普通高等教育"十一五"国家级规划教材



陈明　编著

网站建设实用教程

21世纪计算机科学与技术实践型教程

丛书主编　陈明



清华大学出版社



21 世纪计算机科学与技术实践型教程

丛书主编　陈明

网站建设实用教程

陈　明　编著

清华大学出版社

北　京

内 容 简 介

本书是网站建设的教程，主要内容包括网站设计基础、HTML、CSS 和 JavaScript 基础、ASP. NET 基础、数据库基础与 ADO. NET、远程教育系统实例、Web 服务器和数据库服务器、网站安全和维护等。

本书具有语言精炼、事例丰富、条理清晰、应用性强等特点，既可作为高等院校各专业计算机网站建设的教材，也可作为网站开发人员的参考书。

图书在版编目（CIP）数据

网站建设实用教程/陈明编著. —北京：清华大学出版社，2008.10

（21 世纪计算机科学与技术实践型教程）

ISBN 978-7-302-18055-5

Ⅰ. 网…　Ⅱ. 陈…　Ⅲ. 网站—开发—教材　Ⅳ. TP393.092

中国版本图书馆 CIP 数据核字(2008)第 098211 号

责任编辑： 谢　琛　赵晓宁

责任校对： 白　蕾

责任印制：

出版发行： 清华大学出版社　　**地　　址：** 北京清华大学学研大厦 A 座

http://www.tup.com.cn　　**邮　　编：** 100084

社　总　机： 010-62770175　　**邮　　购：** 010-62786544

投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn

质　量　反　馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn

印 刷 者：

装 订 者：

经　　销： 全国新华书店

开　　本： 185×260　　**印　张：** 22　　**字　数：** 510 千字

版　　次： 2008 年 10 月第 1 版　　**印　次：** 2008 年 10 月第 1 次印刷

印　　数： 1～0000

定　　价： 0.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：010-62770177 转 3103　　产品编号：029947-01

《21世纪计算机科学与技术实践型教程》

编辑委员会

《21 世纪计算机科学与技术实践型教程》

序

21 世纪影响世界的三大关键技术：以计算机和网络为代表的信息技术；以基因工程为代表的生命科学和生物技术；以纳米技术为代表的新型材料技术。信息技术居三大关键技术之首。国民经济的发展采取信息化带动现代化的方针，要求在所有领域中迅速推广信息技术，导致需要大量的计算机科学与技术领域的优秀人才。

计算机科学与技术的广泛应用是计算机学科发展的原动力，计算机科学是一门应用科学。因此，计算机学科的优秀人才不仅应具有坚实的科学理论基础，而且更重要的是能将理论与实践相结合，并具有解决实际问题的能力。培养计算机科学与技术的优秀人才是社会的需要、国民经济发展的需要。

制定科学的教学计划对于培养计算机科学与技术人才十分重要，而教材的选择是实施教学计划的一个重要组成部分，《21 世纪计算机科学与技术实践型教程》主要考虑了下述两方面。

一方面，高等学校的计算机科学与技术专业的学生，在学习了基本的必修课和部分选修课程之后，立刻进行计算机应用系统的软件和硬件开发与应用尚存在一些困难，而《21 世纪计算机科学与技术实践型教程》就是为了填补这部分空白。将理论与实际联系起来，使学生不仅学会了计算机科学理论，而且也学会应用这些理论解决实际问题。

另一方面，计算机科学与技术专业的课程内容需要经过实践练习，才能深刻理解和掌握。因此，本套教材增强了实践性、应用性和可理解性，并在体例上做了改进——使用案例说明。

实践型教学占有重要的位置，不仅体现了理论和实践紧密结合的学科特征，而且对于提高学生的综合素质，培养学生的创新精神与实践能力有特殊的作用。因此，研究和撰写实践型教材是必需的，也是十分重要的任务。优秀的教材是保证高水平教学的重要因素，选择水平高、内容新、实践性强的教材可以促进课堂教学质量的快速提升。在教学中，应用实践型教材可以增强学生的认知能力、创新能力、实践能力以及团队协作和交流表达能力。

实践型教材应由教学经验丰富、实际应用经验丰富的教师撰写。此系列教材的作者不但从事多年的计算机教学，而且参加并完成了多项计算机类的科研项目，他们把积累的经验、知识、智慧、素质融合于教材中，奉献给计算机科学与技术的教学。

我们在组织本系列教材过程中，虽然经过了详细的思考和讨论，但毕竟是初步的尝试，不完善甚至缺陷不可避免，敬请读者指正。

本系列教材主编　陈明

2005 年 1 月于北京

前　　言

网站开发在企业、组织和个人方面都有广泛应用。本书是计算机网站建设的教程，较详细地介绍了网站创建的全过程，主要内容包括网站设计基础、HTML、CSS 和 JavaScript 基础、ASP.NET 基础、数据库基础与 ADO.NET、远程教育系统实例、Web 服务器和数据库服务器、网站安全和维护等。

在网站设计基础中，主要介绍了网站的相关知识、网站的规划和网站的设计等。

在 HTML、CSS 和 JavaScript 基础中，主要介绍了 HTML 语言、CSS 层叠样式表和 JavaScript 语言简介等。

在 ASP.NET 基础中，主要介绍了 Visual Studio.NET 2005、C#基础、ASP.NET 控件和 ASP.NET 基本对象等。

在数据库基础与 ADO.NET 中，主要介绍了数据库基础、ADO.NET 模型简介和 ADO.NET 组件等。

在远程教育系统实例中，主要介绍了系统规划与设计、数据库设计、逻辑分层和功能模块介绍等。

在 Web 服务器和数据库服务器中，主要介绍了 IIS 配置与管理、SQL Server 2000 的配置与管理、发布站点等。

在网站安全和维护中，主要介绍了网站安全、网站的维护等。

学完本书，学生能够广泛地掌握网站的基础知识和开发网站的方法。

本书的主要特点如下。

(1) 内容丰富而广泛。

(2) 注重系统性和科学性，突出了实用性。

(3) 强调培养网站建设的实践能力。

(4) 在编写形式上，力求深入浅出、图文并茂、语言精炼。

由于作者水平有限，书中不足之处在所难免，敬请读者批评指正。

编　者

2008 年 6 月

目　　录

第 1 章　网站设计基础

本章主要讲述网站设计中必须具备的相关知识及网站的规划与设计。通过本章的学习，读者可以了解网站的规划、网站设计的基本思路及网站的设计技巧等相关知识。

1.1　网站的相关知识

随着网络技术的不断发展，网络应用已经渗透到人类社会的各个角落。网站不仅是网络世界的支撑点，更是人们关注的热点：政府利用网站宣传自己的施政纲领，网络已成为政府与百姓交流的直通车；企业利用网站宣传自己的形象，挖掘无限商机；个人利用网站展示个性风采，创建彼此沟通的桥梁。

1.1.1　初识网站

网站是基于 Internet 的，占用一定磁盘空间并提供信息和服务的网络站点，通过网页的形式表现，由 URL(Uniform Resource Location)格式表示。网站可被用来发布信息、宣传企业和开展电子商务等。

一般建设网站有如下 4 个目的。

(1) 信息发布及塑造企业形象。通过 Internet，可发布企业的产品及服务信息，宣传展示企业，塑造企业形象。

(2) 从事商务活动。建设网站，以 Internet 为媒体，充分利用其上的客户群和通信作用进行商务活动。

(3) 吸引投资。根据所建网站拥有的价值出售站点。

(4) 兴趣和爱好。主要是一些个人，因为个人爱好而建设网站。

1.1.2　网站的分类

网站可以从专业角度和内容角度进行分类。

从专业角度可将网站分为专业型网站和个人网站。

(1) 专业型网站：由公司、企业、政府机构和社会团体等经营的专业网站。这类网站含金量高，能够提供多种服务，具有很高的经济和社会价值。

(2) 个人网站：个人建立，以展示自我、介绍自身兴趣、爱好和专长为主要目的的小

型网站。

从内容角度上，网站大致可分为以下几类。

(1) 娱乐与休闲类：如聊天、论坛、棋牌和交友等。

(2) 文学与艺术类：如网络文学、诗歌、纪实传记、文学期刊和小说等。

(3) 计算机与 Internet 类：如软硬件、Internet、电子商务和网页制作等。

(4) 体育与游戏类：如球类运动、健身与健美、养生和计算机游戏等。

(5) 工商经济类：如金融投资、股市、经济热点、房地产和会议展览等。

(6) 公司企业类：如国外公司企业、国内上市公司、机械、电子电气和服装鞋帽等。

(7) 新闻与媒体类：如报纸、杂志、电视、新闻网站和时事论坛等。

(8) 教育与科技类：如留学与移民、考试与入学、远程教育、语言教育和高等教育等。

1.1.3 主页与网页

主页和网页是两个容易混淆的概念。网页泛指 WWW 上所有可供浏览的页面；而主页特指用户进入网站后所看到的第一个页面。当用户在浏览器的地址栏中输入网站的 URL 地址后，浏览器就会自动链接到这个网址所指向的 Web 服务器，打开一个默认的网页作为网站的开始。因此这个总是被最先打开的默认页面称为“主页”或“首页”。

主页是用户登录网站后首先看到的页面，所以主页的设计至关重要。主页应该亮丽美观、表现出网站的风格、内容，吸引用户的注意力，否则很难给用户留下深刻的印象。

1.1.4 网页的组成元素

构成网页的主要元素有文字、图像和超级链接。文字和图像是向用户传导信息的媒介，而超级链接则是链接不同页面的纽带，三部分有机结合，组成了完整的页面。随着网络技术的发展，人们不再满足于简单的静态内容，于是各种多媒体元素也被加入到网页中，如声音、视频图画等。

1. 文字

文字是页面传导信息的主体，页面上的大部分内容都是用文字表现的。文字所占的空间非常小，因此下载速度很快。网页上的文字可以有不同的大小、文体、颜色和格式，从而使网页看起来生动活泼。如果采用一些动态技术，就可以使文字具有各种各样的动态效果。

2. 图像

图像是网页设计必不可少的另一个要素，它能给用户带来更为强烈的视觉效果。图像不但可以直观表达信息，还可以起装饰和美化网页的作用。在页面中经常使用的图像格式有 GIF 和 JPEG 两种。

(1) GIF 图像。GIF(Graphics Interchange Format)是网页中最常用的图像格式。它采用无损压缩方式，图像没有细节上的损失，并且解压打开速度非常快。GIF 图像支持透明背景，能够和网页结合成为一个整体，视觉效果非常协调。另外，GIF 图像还支持动态效果，通过专门软件，如 GIF ANIMATOR 就可以做出动态的 GIF 图像，使网页效果更为

生动。但是 GIF 图像只支持 256 种颜色，所以不适合对颜色质量要求高的场合，如照片等。

（2）JPEG 图像。JPEG（Joint Photographic Expert Group）格式使用先进的压缩算法，使文件比 GIF 图像文件小，因而下载速度快。JPEG 支持高达 1670 万种颜色，非常适合存储颜色丰富的画面，如风景画、照片等。

3. 超级链接

超级链接是 WWW 中最具特色的功能。为了在网页之间自由跳转，可以在网页上的文字或图像中加入目标网页或网站的 URL 地址，这就生成了一个超级链接。当用户在浏览网页时，鼠标经过超级链接时，就会变成小手状态，这时单击，浏览器就会打开超级链接所指的页面。超级链接使页面与页面、网站与网站之间链接成一个整体，用户可以用鼠标自由找到自己想要浏览的网页。

4. 多媒体元素

网页上的多媒体元素通常可以分为音频和动画两种，网页上常用的音频文件有 WAV 和 MIDI 格式；动画文件的格式有 AVI 视频、Java Applet 动画等。还有一些多媒体文件，需要浏览器安装执行插件才可以运行，如 Flash、Generator 和 Shockwave 等生成的文件。

1.2　网站的规划

要开发一个优秀的网站，通常应该遵循以下工作流程：确定开发网站的目的；对网站的外观进行设计；对实际页面进行制作；对所做网站进行测试，以确保它符合最初设计的目标；发布网站。网站发布后还需要对网站进行维护，以便及时更新网站内容。

良好的规划是进一步工作的基础，在网站建设开始时，对网站进行详细设计规划和组织会大大方便以后的工作。网站的规划通常包括以下内容。

1. 确定建立网站的目标

确定建立网站的目标至关重要，如确定建立网站是为了销售产品还是为了做服务；确定网站的对象；确定网站所提供的信息和服务等。

2. 确定网站的类型

目前网站有以下几种类型。

（1）信息：全国性媒体、地方性媒体、商业信息和地方信息。

（2）组织：法律法规数据库、教育站点。

（3）专题：旅游、财经、体育、科学和食品。

（4）个人：个人展台。

3. 确定网站主题内容

在这一阶段，需要掌握一些典型目标用户的基本信息。例如，他们共同的兴趣是什么？他们希望从站点中获得什么？要获得这些信息，既可以做一些问卷调查，也可以从亲

友、同学那里得到一些建议。甚至可以先将网站发布，在站点中设立信息页，从浏览者那里得到实际的信息，然后再对网站进行改进。

4. 确定网站风格

确定网站的整体风格，也就决定了网站内容的表现形式，包括网页所采用的布局结构、颜色、字体、标志图形和图像等。不管采用什么样的风格，只要能够找到文字与图像及其他网络媒体的平衡点，给访问者提供需要的信息，最终吸引访问者，那么网站就是成功的。

Internet 上的网站风格繁多，归纳起来主要有以下两种。

(1) 信息式：这类网站主要以文字信息为主，所以页面的布局要求整齐划一，而且网站中每个层次页面都会有一个导航系统，顶部区域大多使用一些比较有特色的徽标或公司的商标，顶部中间是一些广告横幅，页面剩余部分则分门别类地放置了大量的文本超级链接。整个站点对图像、动画等这些带有修饰成分的多媒体信息采用比较低调的处理，每个页面仅仅包含少量图片或 GIF 动画。目前国内知名的门户网站，如“新浪”、“搜狐”和“网易”等都属于信息式的网站。

(2) 画廊式：这类站点的典型代表是个人网站或公司网站，表现形式主要以图像、动画和多媒体等高新网络技术为主，注重通过各种信息手段表现个人特色或公司要宣扬的理念。

5. 确定网页系统树

系统性对任何一项工作都相当重要，构建一个内容繁多的公司网站绝不能像构建个人网站那样随意增删内容，应该利用系统树对网页进行规划、设计和维护。

6. 确定网页内容

在这里，网页内容并非指网站内容，不是说网页中要包含广告、产品信息和服务等内容，而是要具体细化到每个页面。

7. 定制数据库

数据库对于任何一个网络公司来说是必须具备的，具体可含产品数据库、客户数据库、反馈信息数据库及竞争对手数据库等。

8. 考虑网络的技术问题

在设计网页时要充分考虑网络带宽问题，另外还要考虑浏览器、分辨率及相应的插件等。

1.3 网站的设计

1.3.1 网站设计的基本思路

1. 定位网站的主题和名称

网站的主题也就是网站的题材，它是网站设计时首先遇到的问题。网站题材千奇百

怪，琳琅满目，只要想得到，就可以把它制作出来。定位网站遵循的原则如下。

1）主题要小而精

定位要小，内容要精。如果制作一个包罗万象的站点，把所有人们认为精彩的东西都放在上面，那么往往会事与愿违，因为它给人的感觉是没有主题，没有特色，虽然样样有，却样样都很肤浅，因为人们不可能有那么多的精力去维护它。网站的最大特点就是新和快，目前最热门的个人主页都是天天更新甚至几小时更新一次。最新的调查结果也显示，网络上的“主题站”比“万全站”更受人们喜爱，就好比专卖店和百货商店，如果我需要买某一方面的东西，肯定会选择专卖店。

2）题材最好是自己擅长或者喜爱的内容

兴趣是制作网站的动力，没有热情，很难设计制作出优秀的网站。例如，擅长编程，就可以建立一个编程爱好者网站；对足球感兴趣，可以报道最新的战况、球星动态等。这样在制作时，才不会觉得无聊或者力不从心。

3）题材不要太滥或者目标太高

“太滥”是指到处可见，人人都有的题材，如软件下载、免费信息。“目标太高”是指在这一题材上已经有非常优秀和知名度很高的站点，要超过它是很困难的。

2. 定位网站的 CI 形象

CI(Corporate Identity)是指通过视觉来统一企业的形象。一个杰出的网站，和实体公司一样，需要整体的形象包装和设计。准确的、有创意的 CI 设计，对网站的宣传推广有事半功倍的效果。具体的做法如下。

1）设计网站的标志(logo)

就如同商品的商标一样，网站标志是站点特色和内涵的集中体现。标志的设计创意来自网站的名称和内容。

网站有代表性的人物、动物和花草等，可以用它们作为设计的蓝本，加以卡通化和艺术化，例如迪斯尼的米老鼠、搜狐的卡通狐狸等。

网站有专业性的，可以以本专业有代表的物品作为标志。例如，中国银行的铜板标志、奔驰汽车的方向盘标志等。

最常用和最简单的方式是用自己网站的英文名称作标志。采用不同的字体、字母的变形和字母的组合可以很容易制作好自己的标志。

2）设计网站的标准色彩

网站给人的第一印象来自视觉冲击，确定网站的标准色彩是相当重要的一步。不同的色彩搭配产生不同的效果，并可能影响到访问者的情绪。举个实际的例子就明白了：IBM 的深蓝色、肯德基的红色条型、Windows 视窗标志上的红蓝黄绿色块，都使我们觉得很贴切、很和谐。“标准色彩”是指能体现网站形象和延伸内涵的色彩。一般来说，一个网站的标准色彩不超过三种，太多则让人眼花缭乱。标准色彩要用于网站的标志、标题、主菜单和主色块，它给人以整体统一的感觉。至于其他色彩也可以使用，只是作为点缀和衬托，绝不能喧宾夺主。适合于网页标准色的颜色有蓝色、黄/橙色及黑/灰/白色三大系列色，要注意色彩的合理搭配。

3）设计网站的标准字体

和标准色彩一样，标准字体是指用于网站的标志、标题和主菜单的特有字体。一般网页默认的字体是宋体。为了体现站点的“与众不同”和特有风格，可以根据需要选择一些特别字体。例如，为了体现专业可以使用粗仿宋体，体现设计精美可以用广告体，体现亲切随意可以用手写体等。

4）设计网站的宣传标语

网站的标语是网站的精神和目标。它是用一句话甚至一个词来高度概括。

3. 确定网站的栏目

建立一个网站好比写一篇文章，首先要拟好提纲，文章才能主题明确、层次清晰。如果网站结构不清晰、目录庞杂和内容东一块西一块，结果不但浏览者看得糊涂，自己扩充和维护网站也相当困难。网站的题材确定后，并且收集和组织了许多相关的资料内容，但如何组织内容才能吸引人们浏览网站呢？栏目的实质是一个网站的大纲索引，索引应该将网站的主体明确显示出来。一般的网站栏目安排要注意以下几方面。

1）要紧扣主题

将主题按一定的方法分类并将它们作为网站的主栏目。主题栏目个数在总栏目中要占绝对优势，这样的网站比较专业，主题突出，容易给人留下深刻印象。

2）设立最近更新或网站指南栏目

设立“最近更新”的栏目，是为了照顾常来的访客，让主页更有人性化。如果主页内容庞大，层次较多，而又没有站内的搜索引擎，设置“本站指南”栏目，可以帮助初访者快速地找到他们想要的内容。

3）设立可以双向交流的栏目

如论坛、留言本和邮件列表等，可以让浏览者留下他们的信息。

4）设立下载或常见问题回答栏目

网络的特点是信息共享，如在主页上设置一个资料下载栏目，便于访问者下载所需资料。另外，如果站点经常收到网友关于某方面的问题来信，最好设立一个常见问题回答的栏目，既方便了网友，又可以节省自己更多的时间。

4. 确定网站的目录结构

网站的目录是指建立网站时创建的目录。目录结构的好坏，对浏览者来说并没有什么太大的感觉，但是对于站点本身的上传维护、未来的扩充和移植有着重要的影响。下面是建立目录结构的一些建议。

(1) 不要将所有文件都存放在根目录下，否则会造成文件管理混乱。

所有文件都存放在根目录下，会使用户常常搞不清哪些文件需要编辑和更新，哪些无用的文件可以删除，哪些是相关联的文件，从而影响工作效率。另外，上传速度慢。服务器一般都会为根目录建立一个文件索引。当将所有文件都放在根目录下，那么即使只上传更新一个文件，服务器也需要将所有文件再检索一遍，建立新的索引文件。很明显，文件量越大，等待的时间也将越长。因此，要尽可能地减少根目录的文件存放数。

(2) 按栏目内容建立子目录。

子目录的建立,首先按主菜单栏目建立。例如,企业站点可以按公司简介、产品介绍、价格、在线订单和反馈联系等建立相应目录。其他的次要栏目,类似 what's new,友情链接内容较多,需要经常更新的可以建立独立的子目录。而一些相关性强,不需要经常更新的栏目,如关于本站、关于站长和站点经历等可以合并放在一个统一目录下。所有程序一般都存放在特定目录。例如,CGI 程序放在 cgi-bin 目录。所有需要下载的内容也最好放在一个目录下。

(3) 在每个主栏目目录下都建立独立的 images 目录。

为每个主栏目建立一个独立的 images 目录是最方便管理的。而根目录下的 images 目录只是用来放首页和一些次要栏目的图片。

(4) 目录的层次不要太深。

目录的层次建议不要超过三层,维护管理方便。

(5) 不要使用中文目录。

(6) 不要使用过长的目录。

5. 确定网站的链接结构

网站的链接结构是指页面之间相互链接的拓扑结构。它建立在目录结构基础之上,但可以跨越目录。建立网站的链接结构有两种基本方式。

1) 树状链接结构

类似 DOS 的目录结构,首页链接指向一级页面,一级页面链接指向二级页面。这样的链接结构浏览时,一级级进入,一级级退出。优点是条理清晰,访问者明确知道自己在什么位置,不会"迷"路。缺点是浏览效率低,一个栏目下的子页面到另一个栏目下的子页面,必须绕经首页。

2) 星状链接结构

类似网络服务器的链接,每个页面相互之间都建立有链接。这种链接结构的优点是浏览方便,随时可以到达自己喜欢的页面;缺点是链接太多,容易使浏览者迷路,搞不清自己在什么位置和看了多少内容。

这两种基本结构都只是理想方式,在实际的网站设计中,总是将这两种结构混合起来使用,达到比较理想的效果。比较好的方案是,首页和一级页面之间用星状链接结构,一级和以下各级页面之间用树状链接结构。

6. 设计网站的整体风格

风格(style)是抽象的,是指站点的整体形象给浏览者的综合感受。这个"整体形象"包括站点的 CI(标志、色彩、字体和标语)、版面布局、浏览方式、交互性、文字、语气、内容价值、存在意义和站点荣誉等诸多因素。例如网易是平易近人的,迪斯尼是生动活泼的,IBM 是专业严肃的,这些都是网站给人们留下的不同感受。

风格是独特的,是站点不同于其他网站的地方。或者色彩,或者技术,或者是交互方式,能让浏览者明确分辨出这是网站独有的。例如新世纪网络的黑白色,网易壁纸站的特有框架,即使只看到其中一页,也可以分辨出是哪个网站的。

风格是有人性的。通过网站的外表、内容、文字和交流可以概括出一个站点的个性和情绪。它是温文儒雅,是执著热情,是活泼易变,是放任不羁,就像诗词中的“豪放派”和“婉约派”,则可以用人的性格来比喻站点。如何树立网站风格呢?可以分为以下几个步骤。

(1) 确信风格是建立在有价值内容之上。一个网站有风格而没有内容,就好比绣花枕头一包草;又好比一个性格傲慢但却目不识丁的人。因此,首先必须保证内容的质量和价值性,这是最基本的。

(2) 需要彻底搞清楚希望站点给人的印象是什么。

(3) 在明确网站印象后,开始努力建立和加强这种印象。

经过第二步印象的“量化”后,需要进一步找出其中最有特色特点的东西,就是最能体现网站风格的东西。并以它作为网站的特色加以重点强化、宣传。

1.3.2 网站设计技巧

1. 明确内容

如果想成为一个网站设计者,并正想建一个网站的话,首先应该考虑网站的内容,包括网站功能和用户需要。整个设计都应该围绕这些方面来进行。

2. 抓住用户

如果用户不能够迅速地进入网站,或操作不便捷,网站设计就是失败的。不要让用户失望而转向对手的网站。

3. 快速下载

没有什么比要花很长时间下载页面更糟糕的了。作为一条经验,一个标准的网页应不大于 60KB,通过 56K 调制解调器加载花 30s 的时间。有的设计者说网页加载应在 15s 内。

4. 网站升级

时刻注意网站的运行状况。性能很好的主机随着访问人数的增加,可能会运行缓慢。但为了不失去访问者,一定要仔细计划好网站升级计划。

5. 网站地图

许多设计者把他们的网站地图放在网站上,这种做法却是弊大于利。绝大部分的访问者上网是寻找一些特别的信息,他们对于网站是如何工作的并没有兴趣。如果觉得网站需要地图,那很可能是需要改进网站导航和工具条。

6. 检查错别字

好的拼写是人们一生中重要的技能。但是遗憾的是,许多设计者都缺少这种技能。确保拼写正确,并且格外注意平常容易误写的字。

7. 避免长文本页面

在一个站点上有许多只有文本的页面,是令人乏味的,且也浪费 Web 的潜力。如果网站有大量的基于文本的文档,应当以 Adobe Acrobat 格式的文件形式来放置,以便访问

者能离线阅读。

8. 闪烁让人头痛

通过使用标识可以吸引访问者对网站主页特殊部分的注意，但是这也让访问者头痛。如果想使访问者再次光顾该网站就少用此方法。

9. 背景颜色

背景颜色也会产生一些问题，可能会使网页难于阅读。应当坚持使用白色的背景和黑色的文本，另外还应当坚持使用通用字体。

10. 慎用声音

声音的运用也应得到警惕。内联声音是网页设计者的另一个禁地。因为过多地使用声音会使下载速度很慢，同时并没有带给浏览者多少好处。首次听到鼠标发出声音可能会很有趣，但是多次以后肯定会让人厌烦。使用声音前，应该仔细考虑声音将会带来什么。

1.3.3　网页的布局

布局，就是以最适合浏览的方式将图片和文字排放在页面的不同位置。经常用到的版面布局形式如下。

1. T 结构布局

T 结构，就是指页面顶部为横条网站标志＋广告条，下方左边为主菜单，右边显示内容的布局。因为菜单条背景较深，整体效果类似英文字母 T，所以称为 T 形布局。这是网页设计中用得最广泛的一种布局方式。这种布局的优点是页面结构清晰、主次分明。它是初学者最容易上手的布局方法。缺点是规矩呆板，如果细节色彩上不注意，很容易让人“看之无味”。

2. “口”形布局

这是一个象形的说法，就是页面一般上下各有一个广告条，左边是主菜单，右边放友情链接等，中间是主要内容。这种布局的优点是充分利用版面，信息量大。缺点是页面拥挤，不够灵活。也有将四边空出，只用中间的窗口型设计，例如网易壁纸站。

3. “三”形布局

这种布局多用于国外站点，国内用得不多。其特点是页面上横向两条色块，将页面整体分割为 4 部分，色块中大多放广告条。

4. 对称对比布局

采取左右或者上下对称的布局，一半深色，一半浅色，一般用于设计型站点。它的优点是视觉冲击力强，缺点是将两部分有机地结合比较困难。

5. POP 布局

POP 引自广告术语，就是指页面布局像一张宣传海报，以一张精美图片作为页面的设计中心。常用于时尚类站点，如 ELLE. com。优点是漂亮吸引人；缺点是速度慢。作为版面布局还是值得借鉴的。

1.3.4 网页的设计

在网页设计的认识上，许多人似乎仍停留在网页制作的高度上。认为只要用好了网页制作软件，就能搞好网页设计了。

其实网页设计是一个感性思考与理性分析相结合的复杂的过程，它的方向取决于设计的任务，它的实现依赖于网页的制作。正所谓“功夫在诗外”，网页设计中最重要的东西，并非在软件的应用上，而是在对网页设计的理解及设计制作的水平上，在于我们自身的美感及对页面的把握上。

1. 设计的原则

设计是有原则的，无论使用何种手法对画面中的元素进行组合，都一定要遵循 5 个大的原则：统一、连贯、分割、对比及和谐。

(1) 统一：指设计作品的整体性，一致性。设计作品的整体效果是至关重要的，在设计中切勿将各组成部分孤立分散，那样会使画面呈现出一种枝蔓纷杂的凌乱效果。

(2) 连贯：指要注意页面的相互关系。设计中应利用各组成部分在内容上的内在联系和表现形式上的相互呼应，并注意整个页面设计风格的一致性，实现视觉上和心理上的连贯，使整个页面设计的各个部分极为融洽，犹如一气呵成。

(3) 分割：指将页面分成若干小块，小块之间有视觉上的不同，这样可以使观者一目了然。在信息量很多时为使观者能够看清楚，就要注意到将画面进行有效的分割。分割不仅是表现形式的需要。换个角度来讲，分割也可以被视为对于页面内容的一种分类归纳。

(4) 对比：通过矛盾和冲突，使设计更加富有生气。对比手法很多，例如，多与少、曲与直、强与弱、长与短、粗与细、疏与密、虚与实、主与次、黑与白、动与静、美与丑、聚与散等。在使用对比的时候应慎重，对比过强容易破坏美感，影响统一。

(5) 和谐：指整个页面符合美的法则，浑然一体。如果一件设计作品仅仅是色彩、形状和线条等的随意混合，那么作品将不但没有“生命感”，而且也根本无法实现视觉设计的传达功能。和谐不仅要看结构形式，而且要看作品所形成的视觉效果能否与人的视觉感受形成一种沟通，产生心灵的共鸣。这是设计能否成功的关键。

2. 设计的任务

设计是一种审美活动，成功的设计作品一般都很艺术化。但艺术只是设计的手段，而并非设计的任务。设计的任务是要实现设计者的意图，而并非创造美。

网页设计的任务，是指设计者要表现的主题和要实现的功能。站点的性质不同，设计的任务也不同。从形式上，可以将站点分为以下三类。

(1) 资讯类站点。像新浪、网易和搜狐等门户网站。这类站点将为访问者提供大量的信息，而且访问量较大。因此需注意页面的分割、结构的合理、页面的优化、界面的亲和等问题。

(2) 资讯和形象相结合的网站。像一些较大的公司、国内的高校等。这类网站在设计上要求较高，既要保证资讯类网站的上述要求，同时又要突出企业、单位的形象。然而

就现状来看，这类网站有粗制滥造的嫌疑。

(3) 形象类网站。如一些中小型的公司或单位。这类网站一般较小，有的则有几页，需要实现的功能也较为简单，网页设计的主要任务是突出企业形象。这类网站对设计者的美工水平要求较高。

当然，这只是从整体上来看，具体情况还要具体分析，对不同的站点要区别对待。最重要的一点，那就是客户的要求，它也属于设计的任务。

3. 设计的实现

设计的实现可以分为两个部分。第一部分为站点的规划及草图的绘制，这一部分可以在纸上完成。第二部分为网页的制作，这一过程是在计算机上完成的。

设计首页的第一步是设计版面布局。可以将网页看作传统的报纸杂志来编辑，这里面有文字、图像乃至动画，我们要做的工作就是以最适合的方式将图片和文字排放在页面的不同位置。除了要有一台配置不错的计算机外，软件也是必需的。不能简单地说一个软件的好坏，只要是设计者使用起来觉得方便而且能得心应手的，就可以称为好软件。当然，它应该能满足设计者的要求。常用的软件有 Macromedia 的 Dreamweaver、Fireworks、Flash 以及 Adobe 的 Photoshop、Imageready 等。

接下来要做的就是通过软件的使用，将设计的蓝图变为现实，最终的集成一般是在 Dreamweaver 里完成的。虽然在草图上定出了页面的大体轮廓，但是灵感一般都是在制作过程中产生的。设计作品一定要有创意，这是最基本的要求，没有创意的设计是失败的。

4. 色彩的运用

色彩是一种奇怪的东西，它是美丽而丰富的，能唤起人类的心灵感知。一般来说，红色是火的颜色，热情、奔放；也是血的颜色，可以象征生命。黄色是明度最高的颜色，显得华丽、高贵、明快。绿色是大自然草木的颜色，意味着纯自然和生长，象征安宁和平与安全，如绿色食品。紫色是高贵的象征，有庄重感。白色能给人以纯洁与清白的感觉，表示和平与圣洁。

颜色是光的折射产生的，红、黄、蓝是三原色，其他的色彩都可以用这三种色彩调和而成。换一种思路，可以用颜色的变化来表现光影效果，这无疑将使我们的作品更贴近现实。

色彩代表了不同的情感，有着不同的象征含义。这些象征含义是人们思想交流当中的一个复杂问题，它因人的年龄、地域、时代、民族、阶层、经济地区、工作能力、教育水平、风俗习惯、宗教信仰、生活环境和性别差异而有所不同。

单纯的颜色并没有实际的意义，和不同的颜色搭配，它所表现出来的效果也不同。例如绿色和金黄、淡白搭配，可以产生优雅、舒适的气氛。蓝色和白色混合，能体现柔顺、淡雅和浪漫的气氛。红色和黄色、金色的搭配能渲染喜庆的气氛。设计的任务不同，配色方案也随之不同。考虑到网页的适应性，应尽量使用网页安全色。

但颜色的使用并没有一定的法则，如果一定要用某个法则去套，效果只会适得其反。经验上我们可先确定一种能表现主题的主体色，然后根据具体的需要，应用颜色的近似和对比来完成整个页面的配色方案。整个页面在视觉上应是一个整体，以达到和谐、悦目的

视觉效果。

5. 造型的组合

在网页设计中，主要通过视觉传达来表现主题。在视觉传达中，造型是很重要的一个元素。不管是图还是文字，画面上的所有元素都可以统一作为画面的基本构成要素（点、线、面）来进行处理。在一幅成功的作品里，是需要点、线、面的共同组合与搭配来构造整个页面的。

通常可以使用的组合手法有秩序、比例、均衡、对称、连续、间隔、重叠、反复、交叉、节奏、韵律、归纳、变异、特写和反射等，它们都有各自的特点。在设计中应根据具体情况，选择最适合的表现手法，这样有利于主题的表现。

通过点、线、面的组合，可以突出页面上的重要元素，突出设计的主题，增强美感，让观者在感受美的过程中领会设计的主题，从而实现设计的任务。

造型的巧妙运用不仅能带来极大的美感，而且能较好地突出企业形象，能将网页上的各种元素有机地组织起来，甚至还可以引导观者的视线。

6. 网页的优化

在网页设计中，网页的优化是较为重要的一个环节。它的成功与否会影响页面的浏览速度和页面的适应性，影响观者对网站的印象。

在资讯类网站中，文字是页面中最大的构成元素，因此字体的优化显得尤为重要。使用 CSS 样式表指定文字的样式是必要的，通常将字体指定为宋体，大小指定为 12px，颜色要视背景色而定，原则上以能看清并且与整个页面搭配和谐为准。在白色的背景上，一般使用黑色，这样不易产生视觉疲劳，能保证浏览者较长时间地浏览网页。

图片是网页中的重要元素。图片的优化可以在保证浏览质量的前提下将其 Size 降至最低，这样可以成倍地提高网页的下载速度。利用 Photoshop 或 Fireworks 可以将图片切成小块图片，分别进行优化。输出的格式可以为 GIF 或 JPEG，要视具体情况而定。一般把有较为复杂颜色变化的小块图片优化为 JPEG，而把那种只有单纯色块的卡通画式的小块图片优化为 GIF，这是由这两种格式的特点决定的。

表格（Table）是页面中的重要元素，是页面排版的主要手段。可以设定表格的宽度、高度、边框、背景色和对齐方式等参数。很多时候，将表格的边框设为 0，以此来定位页面中的元素，或者借此确定页面中各元素的相对位置。我们知道，浏览器在读取网页 HTML 原代码时，是读完整个 Table 才将它显示出来的。如果一个大表格中含有多个子表格，必须等大表格读完，才能将子表格一起显示出来。在访问一些站点时，等待多时无结果，单击“停止”按钮却一下显示出页面就是这个原因。因此，在设计页面表格时，应该尽量避免将所有元素嵌套在一个表格里，而且表格嵌套层次尽量要少。在使用 Dreamweaver 制作网页时，会自动在每一个 td 内添加一个空字符“ ”。如果单元格内没有填充其他元素，这个空字符会保留，在指定 td 的宽度或高度后，可以在源代码内将其删去。

网页的适应性是很重要的，在不同的系统、不同的分辨率和不同的浏览器上，将会看到不同的结果，因此设计时要统筹考虑。一般在 800×600 下制作网页，最佳浏览效果也是在 800×600 分辨率下，在其他情况下只要保证基本一致，不出现较大问题即可。

第 2 章　HTML、CSS 和 JavaScript 基础

本章简要介绍构建网页前台页面的主要内容——HTML、CSS 和 JavaScript。通过本章的学习，应该对建设网站的前台脚本有一定的认识。能够自己设计一些简单的静态网页。

超文本标记语言（Hyper Text Mark-up Language，HTML）和一般文本不同，一个 HTML 文件不仅包含文本内容，还包含一些 Tag，中文称“标记”。一个 HTML 文件的后缀是.htm 或者是.html，用文本编辑器就可以编写 HTML 文件。

JavaScript 是一种新的描述语言，此语言可以被嵌入 HTML 的文件之中。透过 JavaScript 可以做到回应使用者的需求事件（如 form 的输入），而不用任何的网络来回传输资料。所以当一位使用者输入一项资料时，它不用经过传给服务端（server）处理，再传回来的过程，而可以直接被客户端（client）的应用程序所处理。

JavaScript 是一种基于对象和事件驱动并具有安全性能的脚本语言。使用它的目的是与 HTML、Java 脚本语言一起实现在一个网页中链接多个对象，与网络客户交互作用，从而可以开发客户端的应用程序。它是通过嵌入或调入在标准的 HTML 语言中实现的。

层叠样式表单（Cascading Style Sheets，CSS）也称作样式表。顾名思义，它是一种设计网页样式的工具。利用其属性可以设置字体、颜色和背景等页面格式；利用其定位可以使页面布局更加规范、好看；利用其滤镜可以使页面产生多媒体效果。CSS 的功能很强大，在后面的教程中将对它进行详细的讲解。

2.1　HTML 语言

2.1.1　网页架构

```
<HTML>
  <HEAD>
    <TITLE>网页制作</TITLE>
      <Meta>…</Meta>
```

```
    </HEAD>
      <BODY>
        BODY 之间则为主要语法所在,也是网页的主要呈现部分。
      </BODY>
</HTML>
```

标签解说:

上面是结构最简单的网页。网页其实就是标签(标签就是指被<>包起来的语法)的集合,透过浏览器的消化整理,就变成了网页。

简单而言,通常一份完整的网页包含了两个部分:抬头(HEAD)和文件本体(BODY)。即上面所看到的<HEAD></HEAD>及<BODY></BODY>。

在抬头部分<HEAD></HEAD>中,有另一组标签<TITLE></TITLE>。打在<TITLE></TITLE>这里面的文字会出现在浏览器视窗最上头蓝色部分里,作为网页的主题。

<HTML></HTML>这组标签是可有可无的。这组标签是告诉浏览器:这是一份 HTML 文件。通常都包在网页的最上下两端,将所有的原始代码都包起来。

2.1.2 分隔标签

1. 文字上的分隔标签

使用方法:

```
强制断行标签<br>、强制分段标签<p>
```

标签解说:在写文章时,有时在特定的地方会强迫断行(
),或是在写完某一段的时候便会分段(<p>)。写网页也一样,而且更需要断行及分段的功能,使整个网页整齐美观。

使用范例如表 2-1 所示。

表 2-1 文字分隔标签

原始代码	呈现结果
这是一个断行的范例 看出来了吗?	这是一个断行的范例看出来了吗?
这是一个分段的范例<p>基本上它会比断行还多空出一行	这是一个分段的范例 基本上分段会比断行还多空出一行

2. 分隔线标签

使用方法:

```
上一段文字内容<hr>下一段文字内容
```

标签解说:利用<hr>标签便可产生一条横分隔线。另外,它的部分属性分别说明如下。

使用范例如表 2-2 所示。

表 2-2　分隔线标签

一般用法	尚未加任何属性
原始代码	普通分隔线<hr>
呈现结果	普通分隔线
颜色属性	用法：<hr color="颜色码或颜色名称">
原始代码	橘色分隔线<hr color="# ff8000">
呈现结果	橘色分隔线
宽度属性	用法：<hr width="宽度">，其单位为 px(像素)，宽度亦可用百分比来作设定，如 50％即意为宽度占屏幕 50％
原始代码	宽度为 240px 的分隔线<hr width= "240">
呈现结果	宽度为 240px 的分隔线
厚度属性	用法：<hr size="厚度">
原始代码	厚度为 5 的分隔线<hr size= "5">
呈现结果	厚度为 5 的分隔线
位置属性	用法：<hr align="水平对齐位置">，其设定值有三个，也就是置左 align＝"left"、置中 align＝"center"和置右 align＝"right"
原始代码	靠右的分隔线<hr align="right">
呈现结果	靠右的分隔线
阴影属性	用法：<hr noshade>，无设定值，只要将 noshade 加入即可，通常会配合颜色设定，效果较佳
原始代码	实心分隔线 (无阴影)<hr noshade>
呈现结果	实心分隔线(无阴影)

2.1.3　排版标签

1. 文字置左、置中、置右

使用方法：刚刚介绍的分段标签＜p＞再加上一些简单的属性设定，就可以让其整个文字段落置左、置中或置右了，如表 2-3 所示。

表 2-3　align 属性

原始代码	呈现结果
文字置左	文字置左
文字置中	文字置中

标签解说：align 是分段标签<p>的属性之一，它的功能是专门设定“水平对齐位置”，其常见的设定值有三个，即置左(align="left")、置中(align="center")和置右(align="right")。

2. 向右缩排标签

使用方法：

```
<blockquote>要缩排的文字</blockquote>
```

标签解说：利用<blockquote></blockquote>标签可以将其包起来的文字，全部往右缩排。而且加一组标签，往右缩排一个单位，加两组标签，往右缩排两个单位，依此类推。

使用范例如表 2-4 所示。

表 2-4 blockquote 属性

原始代码	呈现结果
<blockquote>缩排 1 单位</blockquote>	缩排 1 单位
<blockquote><blockquote>缩排 2 单位 </blockquote></blockquote>	缩排 2 单位

3. 保存原始格式标签

使用方法：

```
<pre>文字内容<pre>
```

标签解说：利用<pre></pre>标签可以将其包起来的文字排版、格式，原封不动地呈现出来。

使用范例如表 2-5 所示。

表 2-5 pre 属性

原始代码	呈现结果
<pre> 文　字 　格　式 </pre>	文　字 　格　式

2.1.4 字体标签

1. 标题标签

使用方法：

```
<h1>标题内容</h1>
```

标签解说：标题的大小一共有 6 种，两个标签一组，也就是<h1>～<h6>，<h1>最大，<h6>最小。使用标题标签时，该标签会将字体变成粗体字，并且会自成一行。

使用范例如表 2-6 所示。

表 2-6　标题标签

原始代码	呈现结果
标题一	标题一
标题二	标题二
标题六	标题六

2. 设定字体大小标签

使用方法：

```
<font size=3>文字内容</font>
```

标签解说：标题的大小一共有 7 种，也就是(最小)～(最大)。另外，还有一种写法"文字内容"。其意思就是说，比预设字大一级。当然，也可以是 font size=+2(比预设字大二级)，或是 font size=-1(比预设字小一级)，一般而言，预设字体多为 3。

使用范例如表 2-7 所示。

表 2-7　字体大小标签

原始代码	呈现结果
字体一或是 字体一	字体一
字体二或是 字体二	字体二
字体七或是 字体七	字体七

3. 字型变化标签

使用方法：

```
<b>文字</b>
```

标签解说：在文字标签里，对于文字的格式也有相当多的变化，如粗体、斜体等，此外，也定义了一些现成的格式供编者使用，如"强调"、"原始代码"等。当然，这只是方便读者参考，并没有强迫说遇到原始代码就要加上"原始代码"的标签。

使用范例如表 2-8 所示。

表 2-8　字型变化标签

原始代码	呈现结果
粗体	**粗体**
<i>斜体</i>	*斜体*

续表

原始代码	呈现结果
<u>底线</u>	底线
^{上标}	上标
_{下标}	下标

4. 文字颜色设定

使用方法：

```
<font color="# fefecb">文字颜色</font>
```

标签解说：文字也可以设定颜色。

使用范例如表 2-9 所示。

表 2-9 文字颜色设定

原始代码	呈现结果
红	红色的字
灰	灰色的字

2.1.5 文字标签

1. 文字字型设定

使用方法：

```
<font face="字型名称">文字</font>
```

标签解说：网页上也可以使用字型。唯一的一个限制是对方也要有该字型，否则看到的仍然还是宋体。另外要说明的是，这个标签并不能保证在每个浏览器上都能正常地显现，看不到特殊的字型时，浏览器仍会以宋体来显示。

使用范例如表 2-10 所示。

表 2-10 文字字型设定

原始代码	呈现结果
楷体_GB2312	楷体_GB2312
华康俪中黑	华康俪中黑

2. 特殊字元

使用方法：

```
 
```

标签解说：很多特殊的符号是需要特别处理的，如＜、＞这两个符号若想要呈现在网

页上是没有办法直接打＜的，要呈现＜必须输入编码表示法“<”，常用的方法如下。

使用范例如表 2-11 所示。

表 2-11　特殊字元

原始代码	呈现结果
	(代表一个不断行空白)
<	＜
>	＞
&	&
"	"

3. 设定文字内定值大小

使用方法：

```
<basefont size="1~7">
```

标签解说：这个标签可以改变文字大小的内定值，直接加在＜body＞标签之后就行了。一般而言，若是没有特别设定，文字大小内定值预定值为 3。

2.1.6　影像标签

1. 影像标签

在使用影像标签时要注意文档名和路径。

使用方法：

```
<img src="boy.gif" alt="本站特约模特儿" align=right border=0 hspace=2 vspace=2 height
=56 width=32>
```

标签解说：目前常见的网页图形格式有两种，就是 gif 及 jpg 格式。gif 格式只有 256 色，不过色彩比较鲜艳干净漂亮，适合计算机美工图案。而 jpg 格式的图案是全彩失真压缩，比较适合一大堆颜色的图片，如照片就较适合用 jpg 格式来呈现。

使用范例如表 2-12 所示。

表 2-12　img 标签中的用法

基本用法	用法：＜img src＝"图档名称、路径"＞ 显示图片最基本的方法(就是不加任何属性)，其中 boy.gif 就是图档的档名
原始代码	<img src= "images/G-FASE4.GIF">嗨！我是本站的模特儿喔！
呈现结果	嗨！我是本站的模特儿喔！
左右间距	用法：＜img src＝"图档" hspace＝"离左右物件的距离"＞ 离文字的水平距离，通常多少也设一点，以免靠文字太近，看起来才不会有太挤的感觉
原始代码	左边的字<img src="images/G-FASE4.GIF" hspace="15">右边的字
呈现结果	左边的字　　右边的字

2. 网页影像重要观念

关于路径：如果路径不对，不管网页名称写得多正确也没用，因为浏览器无法按照路径去找到该有的图片。几种常见的情形如表 2-13 所示。

表 2-13 网页影像

html 档的位置	图档的位置	写法	情形说明
c:\demo	c:\demo	<img src="tad. gif">	图文均在同一目录
c:\demo	c:\demo\images	<img src="images/tad. gif">	图在网页下一层目录
c:\demo	c:\	<img src=" .. /tad. gif">	图在网页上一层
c:\demo	c:\image	<img src=" image/tad. gif">	图文在同一层但不同目录

“../”是回到上一层目录的意思。images/则是进入下一层目录 image 之意，而 images/的意思就是，回到上一层目录，然后再进入目录 images 中。以上使用的均为“相对路径”的概念。

影像单位：或许常常会看到 px 这个单位，这个单位完整的写法是 pixels，称为“像素”。

影像格式：网页用的图档目前只有 gif 格式(即副档名为 gif 的图档，如 bg. gif)及 jpg 格式(即副档名为 jpg 的图档，如 bg. jpg)为一般的浏览器所接受。其他如 bmp 格式或是 pcx 格式都是无法在网页上使用的。

2.1.7 背景标签

1. 背景标签

背景标签只有<BODY>这个标签，其余的效果，只要加上一些简单的属性便可做到。

使用方法：

```
<body bgcolor="#ffffff" background="bg.jpg">
```

标签解说：这个标签是构成网页不可或缺的基本要素之一。背景颜色或图片的设定及链接字体的颜色，都放在<body>标签里面。

2. 内文、链接文字颜色设定

使用方法：

```
<body text="#000000" link="#0000ff" vlink="#ff00ff" alink="#ff0000>
```

标签解说：用字体标签中的颜色属性，可以设定文字的颜色，在<body>标签中，便有设定内文、链接等文字的颜色内定值功能。用法如表 2-14 所示。

表 2-14　设定内文颜色

内文颜色	用法：<body text="颜色码">
原始代码	<html> <head> <title>这是标题</title> </head> <body text="#0906a2"> 这里是本文区 </body> </html>
呈现结果	

使用范例如表 2-15 所示。

表 2-15　设定链接文字

链接颜色	用法：<body link="颜色码"> 设定"链接"的颜色。只要是有链接的地方就会出现指定的颜色，当然，如果按下链接后，那又会变成另一个颜色了，下面会说明
原始代码	<html> <head><title>这是标题</title> </head> <body link="#ff6600"> 链接文字 </body> </html>
呈现结果	

2.1.8 链接标签

1. 链接标签基本概念

链接基本概念：一般而言，链接就是在网页中有些字会有特别的颜色，而且字的底下会有条线，当游标移到那些字上时，会变成手指形状。整理如表 2-16 所示。

表 2-16 相对路径和链接位置

相对路径表示方式	代表链接位置
<a href="text1.htm">	text1.htm 在目前的目录中(就例子而言，就是在 c:\www 中)
<a href="docs/text1.htm">	text1.htm 在名为 docs 的次目录中(就本例而言，就是在 c:\www\docs 中)
<a href="../text1.htm">	text1.htm 在目前目录的上一层目录中(就本例而言，就是在 c:\底下)

2. 网页内部的链接

使用方法：先在欲链接处作记号这里是想链接的点。

设定链接：链接

标签解说：当某页的内容很多时，可以利用网页的内部链接，让使用者快速找到资料。

使用范例如表 2-17 所示。

表 2-17 网页的链接步骤

范例	第 1 步	第 2 步
www 链接标签基本概念	欲链接的位置	www 链接标签基本概念
网页内部的链接	欲链接的位置	网页内部的链接
网页外部的链接	欲链接的位置	网页外部的链接

3. 网页外部的链接

标签解说：链接到外面去，可以扩充网站的实用性及充实性。由于网络上的服务内容丰富，所以不同的服务有不同的链接方法，将之整理如下。

使用范例如表 2-18 所示。

表 2-18 外部网的链接

网站链接	好站	好站
电子邮件链接	写信给我	写信给我
ftp 链接	下载档案	下载档案

续表

news 链接	seednet news 服务	seednet news 服务
gopher 链接	seednet gopher 服务	seednet gopher 服务
bbs 链接	seednet bbs 服务	seednet bbs 服务

4. 链接标签的参数

使用方法：在链接后面加入 target=_参数。

标签解说：链接的参数并不多，常见的大概就属 target 这个参数了。target 的意思是“目标”，也就是网页链接的指向目标，该参数在框窗(frame)里尤为重要。

使用范例如下。

- target=框窗名称：这在“框架标签”中也有提到，而且也只有在框架(框窗 or frame)中才用得到。正常而言，框窗有各自的名称，因此，可以利用此标签，来指定链接的内容显示到哪一个框窗中。
- target=_blank：将链接的画面内容，开在新的浏览视窗中。
- target=_parent：将链接的画面内容，当成文件的上一个画面。
- target=_self：将链接的画面内容，显示在目前的视窗中。
- target=_top：这个参数可以解决新链接的画面内容被旧框窗包围的困扰，使用这个参数，会将整个画面重新显示成链接的画面内容。

2.1.9　表格标签

1. 网页中的表格观念

范例如表 2-19 所示。

表 2-19　在表格中设置一个格子

原始代码	呈现结果
<TABLE BORDER=1> <TR><TD>1</TD></TR></TABLE>	1

利用<TABLE>标签来告诉计算机，这是一个表格，至于 BORDER=1 这个参数是设定此表格的框线粗细为 1。一组<TR></TR>是设定一横列的开始，一组<TD></TD>则是设定一个栏位。现在再来增加两个格子，如表 2-20 所示。

表 2-20　在表格中增加两个格子

原始代码	呈现结果
<TABLE BORDER=1> <TR><TD>1</TD><TD>2</TD><TD>3</TD></TR> </TABLE>	1 2 3

2. 表格栏位对齐位置设定

现在制作一个宽 100、高 60 的表格，做法如表 2-21 所示。

表 2-21　左对齐

原始代码	呈现结果
<TABLE WIDTH="100" BORDER="1" HEIGHT="60"> <TR><TD>1</TD></TR></TABLE>	1

在<TD>加入 ALIGN=CENTER 参数，即可得到 1 居中的结果，如表 2-22 所示。

表 2-22　居中

原始代码	呈现结果
<TABLE WIDTH="100" BORDER="1" HEIGHT="60"> <TR><TD ALIGN=CENTER>1</TD></TR></TABLE>	1

此外，利用 ALIGN=RIGHT 可以让表格中物件置右、利用 ALIGN=LEFT 可以让表格中物件置左（预设值）。

利用 VALIGN=MIDDLE 可以让表格中物件垂直置中（预设值），VALIGN=BOTTOM 可以让表格中物件靠下方对齐。利用 VALIGN=TOP 这种属性即可让表格内物件靠上方对齐。

3. 表格背景、底色设定

利用 BGCOLOR="颜色码"就可以改变表格的底色。表 2-23 介绍指定整格表格背景颜色的方法。

表 2-23　设置表格背景颜色

原始代码	呈现结果
<TABLE BORDER="1" BGCOLOR=#FFCC33> <TR><TD>1</TD><TD>2</TD></TR> <TR><TD>3</TD><TD>4</TD></TR> </TABLE>	1 2 3 4

将 BGCOLOR="颜色码"加在<TR>中，可以指定一行的背景颜色，如表 2-24 所示。

表 2-24　设置一行的背景颜色

原始代码	呈现结果
<TABLE BORDER="1"> <TR BGCOLOR=#FFCC33><TD>1</TD> <TD>2</TD></TR><TR><TD>3</TD><TD>4</TD> </TR></TABLE>	1 2 3 4

将 BGCOLOR="颜色码"加在<TD>中，可以指定一格的背景颜色，如表 2-25 所示。

表 2-25　设置一格的背景颜色

原始代码	呈现结果
<TABLE BORDER="1"><TR> <TD BGCOLOR=#FFCC33>1</TD><TD>2</TD></TR> <TR><TD>3</TD><TD>4</TD></TR></TABLE>	1 2 3 4

表格除了可以设定底色外，也可以用图片来作背景。只要将 BACKGROUND="图片名称"加到表格中就行了。和表格背景颜色一样，不但表格可以设定背景图片，也可以指定某栏或某列的背景图片。

将 BACKGROUND="图片名称"加在<TD>中，可以指定一栏的背景颜色，具体方法见下例。

4. 表格框线设定

利用 BORDER="粗细值"可以设定表格粗细，如表 2-26 所示。

表 2-26　设置表格边框粗细

原始代码	呈现结果
<TABLE BORDER=5><TR><TD>1</TD></TR></TABLE>	1

利用 BORDERCOLOR="颜色码"可以设定表格颜色，如表 6-27 所示。

表 6-27　设置表格颜色

原始代码	呈现结果
<TABLE BORDER="5" BORDERCOLOR="#0080FF"> <TR><TD>1</TD></TR></TABLE>	1

另外，利用 BORDERCOLORLIGHT="#颜色码"(亮面设定) BORDERCOLOR-DARK="颜色码"(暗面设定)可以设定表格的阴影、亮面颜色，如表 2-28 所示。

表 2-28　设置表格的阴影、亮面颜色

原始代码	呈现结果
<TABLE BORDER="5" BORDERCOLOR="#0080FF" BORDERCOLORLIGHT="#62B0FF" BORDERCOLORDARK="#004B97"> <TR><TD>1</TD></TR> </TABLE>	1

5. 表格栏距设定

利用 CELLPADDING 属性自由设定表格内文距离格线的距离，如表 2-29 所示。

表 2-29　设置表格内文距离格线的距离

原始代码	呈现结果
<TABLE BORDER="1" CELLPADDING="10"> <TR><TD>1</TD><TD>2</TD></TR></TABLE>	1 2

利用 CELLSPACING 属性设定表格栏位格线之间的距离，如表 2-30 所示。

表 2-30　设置表格栏位格线之间的距离

原始代码	呈现结果
<TABLE BORDER="1" CELLSPACING="5"> <TR><TD>1</TD><TD>2</TD></TR></TABLE>	1 2

2.1.10　序列标签

1. 无序标签

序列标签基本上可分为两种，一种是"无序条列"，一种是"有序条列"。"无序条列"就是指各条列间并无顺序关系，纯粹只是利用条列式方法来呈现资料而已，此种无序标签，在各条列前面均有一符号以示区隔。至于"有序条列"就是指各条列之间是有顺序的，从1、2、3、…一直延伸下去。

首先介绍 "无序列表标签"的使用，如表 2-31 所示。

表 2-31　无序标签的使用方法

原始代码	呈现结果
<UL><LI>姓名：杰出 <LI>生日：1974/11/21 <LI>星座：天蝎座 </UL>	○　姓名：杰出 ○　生日：1974/11/21 ○　星座：天蝎座

其中<UL>标签即为"无序列表标签"，每增加一列内容，就必须加一个<LI>。

前面的符号可以通过 TYPE="形状名称"属性来改变其符号形状，一共有三个选择：DISK(实心圆)、SQUARE(小正方形)和 CIRCLE(空心圆)。

2. 有序标签

首先介绍"有序列表标签"的使用，如表 2-32 所示。

表 2-32　有序标签的使用方法

原始代码	呈现结果
<OL><LI>姓名：杰出 <LI>生日：1974/11/21 <LI>星座：天蝎座 </OL>	1.　姓名：杰出 2.　生日：1974/11/21 3.　星座：天蝎座

其中<OL>标签即为"有序列表标签"，每增加一列内容，就必须加一个<LI>。和

无序列表标签一样,也可以选择不同的符号来显示顺序,一样是用 TYPE 属性来作更改。一共有 5 种符号：1(数字)、A(大写英文字母)、a(小写英文字母)、I(大写罗马字母)和 i(小写罗马字母)。

可指定序列起始的数目,其方法如表 2-33 所示。

表 2-33　指定起始序列的方法

原始代码	呈现结果
<OL START="8"> <LI>姓名：杰出 <LI>生日：1974/11/21 <LI>星座：天蝎座 </OL>	8.　姓名：杰出 9.　生日：1974/11/21 10.　星座：天蝎座

2.1.11　表单标签

1. 各种输入类型

文字输入列：每个表单之所以会有不同的类型,原因就在于 TYPE="表单类型"设定的不同而已。首先来看看第一个类型：文字输入列。文字输入列的形态就是 TYPE="TEXT",其使用方法如表 2-34 所示。

表 2-34　文字输入列的使用方法

呈现结果	窗体顶端 姓名：[　　　　] 窗体底端
原始代码	<FORM> 姓名：<INPUT TYPE="TEXT" NAME="NAME" SIZE="20"> </FORM>

其有下列可设定的属性。

- NAME="名称"：设定此栏位的名称,程式中常会用到。
- SIZE="数值"：设定此栏位显现的宽度。
- VALUE="预设内容"：设定此栏位的预设内容。
- ALIGN="对齐方式"：设定此栏位的对齐方式,其值有 TOP(向上对齐)、MIDDLE(向中对齐)、BOTTOM(向下对齐)、RIGHT(向右对齐)、LEFT(向左对齐)、TEXTTOP(向文字顶部对齐)、BASELINE(向文字底部对齐)、ABSMIDDLE(绝对置中)和 ABSBOTTOM(绝对置下)等。
- MAXLENGTH="数值"：设定此栏位可设定输入的最大长度。

单选核取表单：利用 TYPE="RADIO"就会产生单选核取表单。单选核取表单通常是好几个选项一起摆出来供使用者点选,一次只能从中选一个,故为单选核取表单,如表 2-35 所示。

表 2-35 单选核取表单的使用方法

呈现结果	窗体顶端 性别：男○ 女○ 窗体底端
原始代码	<FORM>性别： 男<INPUT TYPE="RADIO" NAME="SEX" VALUE="BOY">女 <INPUT TYPE="RADIO" NAME="SEX" VALUE="GIRL"> </FORM>

其有下列可设定的属性。

- NAME="名称"：设定此栏位的名称，程式中常会用到。
- VALUE="内容"：设定此栏位的内容、值或是意义。
- ALIGN="对齐方式"：设定此栏位的对齐方式，其值有 TOP(向上对齐)、MIDDLE(向中对齐)、BOTTOM(向下对齐)、RIGHT(向右对齐)、LEFT(向左对齐)、TEXTTOP(向文字顶部对齐)、BASELINE(向文字底部对齐)、ABSMIDDLE(绝对置中)和 ABSBOTTOM(绝对置下)等。
- CHECKED：设定此栏位为预设选取值。

复选核取表单：利用 TYPE=" CHECKBOX "就会产生复选核取表单。复选核取表单通常是好几个选项一起摆出来供使用者点选，一次可以同时选好几个，故为复选核取表单如表 2-36 所示。

表 2-36 复选核取表单的使用方法

呈现结果	窗体顶端 喜好：□ 电影□ 看书 窗体底端
原始代码	<FORM>喜好： <INPUT TYPE="CHECKBOX" NAME="SEX" VALUE="MOVIE">电影 <INPUT TYPE="CHECKBOX" NAME="SEX" VALUE="BOOK">看书 </FORM>

其有下列可设定的属性。

- NAME="名称"：设定此栏位的名称，程式中常会用到。
- VALUE="内容"：设定此栏位的内容、值或是意义。
- ALIGN="对齐方式"：设定此栏位的对齐方式，其值有 TOP(向上对齐)、MIDDLE(向中对齐)、BOTTOM(向下对齐)、RIGHT(向右对齐)、LEFT(向左对齐)、TEXTTOP(向文字顶部对齐)、BASELINE(向文字底部对齐)、ABSMIDDLE(绝对置中)和 ABSBOTTOM(绝对置下)等。
- CHECKED：设定此栏位为预设选取值。

密码表单：利用 TYPE=" PASSWORD "就会产生一个密码表单。密码表单和文字输入表单长得几乎一样，差别就在于密码表单在输入时全部会以星号来取代输入的文字，以防他人偷窥，如表 2-37 所示。

表 2-37　密码表单的使用方法

呈现结果	窗体顶端
	请输入密码：[ ]
	窗体底端
原始代码	<FORM> 请输入密码： <INPUT TYPE="PASSWORD" NAME="INPUT"> </FORM>

其有下列可设定的属性。

- NAME="名称"：设定此栏位的名称，程式中常会用到。
- SIZE="数值"：设定此栏位显现的宽度。
- VALUE="预设内容"：设定此栏位的预设内容，不过呈现出来仍是星号。
- ALIGN="对齐方式"：设定此栏位的对齐方式，其值有 TOP(向上对齐)、MIDDLE(向中对齐)、BOTTOM(向下对齐)、RIGHT(向右对齐)、LEFT(向左对齐)、TEXTTOP(向文字顶部对齐)、BASELINE(向文字底部对齐)和 ABSMIDDLE(绝对置中)等。
- MAXLENGTH="数值"：设定此栏位可设定输入的最大长度。

送出按钮：通常在表单填完之后，都会有一个送出按钮及清除重写的按钮，分别是利用 TYPE="SUBMIT"及 TYPE="RESET"，如表 2-38 所示。

表 2-38　送出按钮的使用方法

呈现结果	窗体顶端
	[送出资料] [重新填写]
	窗体底端
原始代码	<FORM> <INPUT TYPE="SUBMIT" VALUE="送出资料"> <INPUT TYPE="RESET" VALUE="重新填写"> </FORM>

其有下列可设定的属性。

- NAME="名称"：设定此按钮的名称。
- VALUE="文字"：设定此按钮上要呈现的文字，若是没有设定，浏览器也会自动替用户加上“送出查询”、“重设”等字样。
- ALIGN="对齐方式"：设定此栏位的对齐方式，其值有 TOP(向上对齐)、MIDDLE(向中对齐)、BOTTOM(向下对齐)、RIGHT(向右对齐)、LEFT(向左对齐)、

TEXTTOP(向文字顶部对齐)、BASELINE(向文字底部对齐)、ABSMIDDLE(绝对置中)和 ABSBOTTOM(绝对置下)等。

按钮元件：表单中或是 JavaScript 常会用到按钮来作一些效果，因此，可以利用 TYPE="BUTTON"来产生一个按钮，如表 2-39 所示。

表 2-39 按钮元件的使用方法

呈现结果	窗体顶端
	请按下按钮：我同意
	窗体底端
原始代码	<FORM>请按下按钮： <INPUT TYPE="BUTTON" NAME="OK" VALUE="我同意"> </FORM>

其有下列可设定的属性。

- NAME="名称"：设定此按钮的名称。
- VALUE="文字"：设定此按钮上要呈现的文字。
- ALIGN="对齐方式"：设定此栏位的对齐方式，其值有 TOP(向上对齐)、MIDDLE(向中对齐)、BOTTOM(向下对齐)、RIGHT(向右对齐)、LEFT(向左对齐)、TEXTTOP(向文字顶部对齐)、BASELINE(向文字底部对齐)、ABSMIDDLE(绝对置中)和 ABSBOTTOM(绝对置下)等。

隐藏栏位：表单中有时有些东西因为某些因素，不想让使用者看到，但因程式需要却又不得不存在，可以利用 TYPE=" HIDDEN "来产生一个隐藏的栏位，如表 2-40 所示。

表 2-40 隐藏栏位的使用方法

呈现结果	窗体顶端
	隐藏栏位：看不到
	窗体底端
原始代码	<FORM>隐藏栏位： <INPUT TYPE="HIDDEN" NAME="NOSEE" VALUE="看不到"> </FORM>

其有下列可设定的属性。

- NAME="名称"：设定此栏位的名称。
- VALUE="文字"：设定此栏位的值、文字或意义。

2. 大量文字输入元件

如果需要使用者输入比较大量的文字，可以利用<TEXTAREA></TEXTAREA>来产生一个可以输入大量文字的元件，夹在两个标签中的文字会出现在列表框中，可作为预设文字，如表 2-41 所示。

表 2-41　大量文字输入元件的使用方法

呈现结果	窗体顶端
	请输入您的意见： 窗体底端
原始代码	<FORM>请输入您的意见： <TEXTAREA NAME="TALK" COLS="20" ROWS="3"></TEXTAREA> </FORM>

其有下列可设定的属性。

- NAME="名称"：设定此栏位的名称。
- WRAP="设定值"：设定此栏位的换行模式。设定值有三种：OFF(输入文字不会自动换行)、VIRTUAL(输入文字在屏幕上会自动换行，不过若是使用者没有自行按 Enter 键换行，送出资料时，也视为没有换行)和 PHYSICAL(输入文字会自动换行，送出资料时，会将屏幕上的自动换行，视为换行效果送出)。
- COLS="数值"：设定此栏位的行数(横向字数)。
- ROWS="数值"：设定此栏位的列数(垂直字数)。

3. 下拉式列表

利用<SELECT NAME="名称">可以生成一个下拉式列表，另外，还需要配合<OPTION>标签来产生选项，如表 2-42 所示。

表 2-42　下拉式列表的使用方法

呈现结果	窗体顶端
	您喜欢看书吗?：非常喜欢
	窗体底端
原始代码	<FORM> 您喜欢看书吗?： <SELECT NAME="LIKE"> <OPTION VALUE="非常喜欢">非常喜欢 <OPTION VALUE="还算喜欢">还算喜欢 <OPTION VALUE="不太喜欢">不太喜欢 <OPTION VALUE="非常讨厌">非常讨厌 </SELECT> </FORM>

其有下列可设定的属性。

- SIZE="数值"：设定此栏位的大小，预设值为 1。
- MULTIPLE：设定此栏位为复选，可以一次选好几个选项。

2.1.12　框架标签

1. 框架的基本概念

以图 2-1 为例，一共分为 1、2、3 三个框架，每一个框架，各有其显示的内容，分别是 a. htm、b. htm 和 c. htm 三个档案。所有 Frame 的标签，其实都只摆在 index. htm 这个档案里。

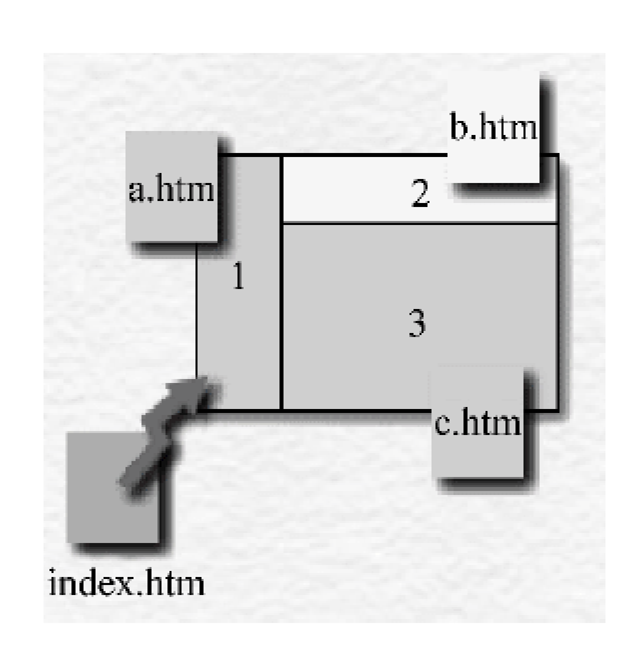


图 2-1　框架示意图

2. 使用框架标签

创建一个网页，如表 2-43 所示。

表 2-43　创建一个网页

原始代码	呈现结果
<HTML><HEAD> <TITLE>框窗实作</TITLE> </HEAD></HTML>	

将画面分成左右两边(左边就是框窗 1 了，右边就暂定为 2)，如表 2-44 所示。

表 2-44　创建 2 个网页

原始代码	呈现结果
<HTML><HEAD> <TITLE>框窗实作</TITLE></HEAD> <FRAMESET COLS="120, * " > <FRAME SRC="a.htm" NAME="1"> <FRAME SRC="b.htm" NAME="2"> </FRAMESET></HTML>	

在<FRAMESET>中，一开始是左右分，所以写成<FRAMESET COLS="120，* " >。COLS="120，* " 就是说，左边那一栏强制定为 120 点，右边则随视窗大小而变。除了直接写点数外，亦可用百分比来表示，例如 COLS="20%，80%"也是可以的。

再将右边的框窗割成上下两个画面(也就是框窗 2 和 3)，如表 2-45 所示。

表 2-45　进行框窗的分割

原始代码	呈现结果
<HTML><HEAD> <TITLE>框窗实作</TITLE> </HEAD> <FRAMESET COLS="120, * "> <FRAME SRC="a.htm" NAME="1"> <FRAMESET ROWS="100, * "> <FRAME SRC="b.htm" NAME="2"> <FRAME SRC="c.htm" NAME="3"> </FRAMESET> </FRAMESET></HTML>	

原本的<FRAME SRC="b. htm" NAME="2">(在第 3 点的语法中)被另一组<FRAMESET ROWS="100, * ">所取代了,第二组<FRAMESET ROWS="100, * ">是被第一组<FRAMESET COLS="120, * ">所包围起来的。

2.2　CSS 层叠样式表

2.2.1　认识 CSS

CSS 是一种制作网页的新技术。网页设计最初是用 HTML 标记来定义页面文档及格式,例如标题<h1>、段落<p>、表格<table>和链接<a>等,但这些标记不能满足更多的文档样式需求。为了解决这个问题,在 1997 年 W3C(The World Wide Web Consortium)颁布 HTML4 标准的同时也公布了有关样式表的第一个标准 CSS1。自 CSS1 的版本之后,又在 1998 年 5 月发布了 CSS2 版本,样式表得到了更多的充实。W3C 把 DHTML(Dynamic HTML)分为三个部分来实现:脚本语言(包括 JavaScript、VBScript 等)、支持动态效果的浏览器(包括 Internet Explorer、Netscape Navigator 等)和 CSS 样式表。

2.2.2　CSS 语法

1. 基本语法

CSS 的定义是由三个部分构成:选择符(selector)、属性(properties)和属性的取值(value)。

基本格式如下:

```
selector {property: value}
```

选择符{属性:值}。选择符可以是多种形式,一般是要定义样式的 HTML 标记,例如 BODY、P 和 TABLE 等,可以通过此方法定义它的属性和值,属性和值要用冒号隔开。

```
body {color: black}
```

选择符 body 是指页面主体部分，color 是控制文字颜色的属性，black 是颜色的值，此例的效果是使页面中的文字为黑色。

如果属性的值是多个单词组成，必须在值上加引号，例如字体的名称经常是几个单词的组合：

```
p {font-family: "sans serif"}
```

（定义段落字体为 sans serif）

如果需要对一个选择符指定多个属性时，使用分号将所有的属性和值分开。

```
p {text-align: center; color: red}
```

（段落居中排列，并且段落中的文字为红色）

为了使自定义的样式表方便阅读，可以采用分行的书写格式。

```
p{text-align: center;
color: black;
font-family: arial
}
```

（段落排列居中，段落中文字为黑色，字体是 arial）

2. 选择符组

可以把相同属性和值的选择符组合起来书写，用逗号将选择符分开，这样可以减少样式重复定义：h1，h2，h3，h4，h5，h6 { color：green }（这个组里包括所有的标题元素，每个标题元素的文字都为绿色）。

```
p, table{ font-size: 9pt }
```

（段落和表格里的文字尺寸为 9 号字）

效果完全等效于：

```
p { font-size: 9pt }
table { font-size: 9pt }
```

3. 类选择符

用类选择符能够把相同的元素分类定义不同的样式，定义类选择符时，在自定类的名称前面加一个点号。假如想要两个不同的段落，一个段落向右对齐，一个段落居中，可以先定义两个类：

```
p.right {text-align: right}
p.center {text-align: center}
```

然后用在不同的段落里,只要在 HTML 标记里加入自定义的 class 参数,如:

```
<p class="right">
这个段落向右对齐的
</p>
<p class="center">
这个段落是居中排列的
</p>
```

注意:类的名称可以是任意英文单词或以英文开头与数字的组合,一般以其功能和效果简要命名。

类选择符还有一种用法,在选择符中省略 HTML 标记名,这样可以把几个不同的元素定义成相同的样式。

```
.center {text-align: center}
```

(定义.center 的类选择符为文字居中排列)

这样的类可以被应用到任何元素上。下面使 h1 元素(标题 1)和 p 元素(段落)都归为 center 类,这使两个元素的样式都跟随.center 这个类选择符。

```
<h1 class="center">
这个标题是居中排列的
</h1>
<p class="center">
这个段落也是居中排列的
</p>
```

注意:这种省略 HTML 标记的类选择符是最常用的 CSS 方法,使用这种方法可以很方便地在任意元素上套用预先定义好的类样式。

4. ID 选择符

在 HTML 页面中 ID 参数指定了某个单一元素,ID 选择符是用来对这个单一元素定义单独的样式。

ID 选择符的应用和类选择符类似,只要把 CLASS 换成 ID 即可。将上例中类用 ID 替代:

```
<p id="intro">
这个段落向右对齐
</p>
```

定义 ID 选择符要在 ID 名称前加上一个“#”号。和类选择符相同，定义 ID 选择符的属性也有两种方法。下面这个例子，ID 属性将匹配所有 id="intro"的元素。

```
#intro
{font-size:110%;
font-weight:bold;
color:#0000ff;
background-color:transparent
}
```

(字体尺寸为默认尺寸的 110%，粗体，蓝色，背景颜色透明)

下面这个例子，ID 属性只匹配 id="intro"的段落元素。

```
p#intro
{font-size:110%;
font-weight:bold;
color:#0000ff;
background-color:transparent
}
```

注意：ID 选择符局限性很大，只能单独定义某个元素的样式，一般只在特殊情况下使用。

5. 包含选择符

可以单独对某种元素包含关系定义的样式表，元素 1 里包含元素 2，这种方式只对在元素 1 里的元素 2 定义，对单独的元素 1 或元素 2 无定义，例如：

```
table a
{font-size: 12px
}
```

在表格内的链接改变了样式，文字大小为 12 像素，而表格外链接的文字仍为默认大小。

6. 样式表的层叠性

层叠性就是继承性，样式表的继承规则是外部的元素样式会保留下来继承给这个元素所包含的其他元素。事实上，所有在元素中嵌套的元素都会继承外层元素指定的属性值，有时会把很多层嵌套的样式叠加在一起，除非另外更改。例如在 DIV 标记中嵌套 P 标记：

```
div { color: red; font-size:9pt}
⋮
<div>
```

```
<p>
这个段落的文字为红色 9 号字
</p>
</div>
```

(P 元素里的内容会继承 DIV 定义的属性)

注意：有些情况下内部选择符不继承周围选择符的值，但理论上这些都是特殊的。例如，上边界属性值是不会继承的，一个段落不会有同文档 BODY 一样的上边界值。

另外，当样式表继承遇到冲突时，总是以最后定义的样式为准。如果上例中定义了 P 的颜色：

```
div { color: red; font-size:9pt}
p {color: blue}
⋮
<div>
<p>
这个段落的文字为蓝色 9 号字
</p>
</div>
```

可以看到段落里的文字大小为 9 号字，是继承 div 属性的，而 color 属性则依照最后定义的。

不同的选择符定义相同的元素时，要考虑到不同选择符之间的优先级。ID 选择符、类选择符和 HTML 标记选择符中，因为 ID 选择符是最后加在元素上的，所以优先级最高，其次是类选择符。如果想超越这三者之间的关系，可以用!important 提升样式表的优先权，例如：

```
p { color: #FF0000!important }
.blue { color: #0000FF}
#id1 { color: #FFFF00}
```

同时对页面中的一个段落加上这三种样式，它最后会依照被!important 申明的 HTML 标记选择符样式为红色文字。如果去掉!important，则依照优先权最高的 ID 选择符为黄色文字。

7. 注释

可以在 CSS 中插入注释来说明代码的意思，注释有利于以后编辑和更改代码时理解代码的含义。在浏览器中，注释是不显示的。CSS 注释以"/*"开头，以"*/"结尾，如下：

```
/* 定义段落样式表 */
p{
text-align: center;            /* 文本居中排列 */
color: black;                  /* 文字为黑色 */
```

```
font-family: arial                /* 字体为 arial */
}
```

2.2.3 如何在网页中插入 CSS

前面了解了 CSS 的语法，但要想在浏览器中显示出效果，就要让浏览器识别并调用。当浏览器读取样式表时，要依照文本格式来读，这里介绍 4 种在页面中插入样式表的方法：链入外部样式表、内部样式表、导入外部样式表和内嵌样式。

1. 链入外部样式表

链入外部样式表是把样式表保存为一个样式表文件，然后在页面中用<link>标记链接到这个样式表文件，这个<link>标记必须放到页面的<head>区内，如下：

```
<head>
⋮
<link rel="stylesheet" type="text/css" href="mystyle.css">
⋮
</head>
```

上面这个例子表示浏览器从 mystyle.css 文件中以文档格式读出定义的样式表。rel="stylesheet"是指在页面中使用这个外部的样式表。type="text/css"是指文件的类型是样式表文本。href="mystyle.css"是文件所在的位置。

一个外部样式表文件可以应用于多个页面。当改变这个样式表文件时，所有页面的样式都随之而改变。在制作大量相同样式页面的网站时，它非常有用，不仅减少了重复的工作量，而且有利于以后的修改、编辑，浏览时也减少了重复下载代码。

样式表文件可以用任何文本编辑器(如记事本)打开并编辑，一般样式表文件扩展名为.css。内容是定义的样式表，不包含 HTML 标记。mystyle.css 文件的内容如下：

```
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
```

(定义水平线的颜色为土黄；段落左边的空白边距为 20 像素；页面的背景图片为 images 目录下的 back40.gif 文件)

2. 内部样式表

内部样式表是把样式表放到页面的<head>区里，这些定义的样式就应用到页面中了，样式表是用<style>标记插入的，从下例中可以看出<style>标记的用法。

```
<head>
⋮
```

```
<style type="text/css">
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
</style>
⋮
</head>
```

注意：有些低版本的浏览器不能识别 style 标记，这意味着低版本的浏览器会忽略 style 标记里的内容，并把 style 标记里的内容以文本直接显示到页面上。为了避免这样的情况发生，用加 HTML 注释的方式（＜！——注释——＞）隐藏内容而不让它显示。

```
<head>
⋮
<style type="text/css">
<!--
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
-->
</style>
⋮
</head>
```

3. 导入外部样式表

导入外部样式表是指在内部样式表的＜style＞里导入一个外部样式表，导入时用@import。看下面这个实例：

```
<head>
⋮
<style type="text/css">
<!--
@import "mystyle.css"
其他样式表的声明
-->
</style>
⋮
</head>
```

例中@import "mystyle.css"表示导入 mystyle.css 样式表，注意使用时外部样式表的路径。方法和链入样式表的方法很相似，但导入外部样式表输入方式更有优势。实质上它相当于存在内部样式表中的。

注意：导入外部样式表必须在样式表的开始部分，在其他内部样式表上面。

4. 内嵌样式

内嵌样式是混合在 HTML 标记里使用的，用这种方法，可以很简单地对某个元素单独定义样式。内嵌样式的使用是直接在 HTML 标记里加入 style 参数。而 style 参数的内容就是 CSS 的属性和值，如下例：

```
<p style="color: sienna; margin-left: 20px">
这是一个段落
</p>
```

（这个段落颜色为土黄，左边距为 20 像素）

在 style 参数后面引号里的内容相当于在样式表大括号里的内容。

注意：style 参数可以应用于任意 BODY 内的元素（包括 BODY 本身），除了 BASEFONT、PARAM 和 SCRIPT。

5. 多重样式表的叠加

上一节里已经提到样式表的层叠顺序，已经讨论插入样式表的这几种方法的叠加。如果在同一个选择器上使用几个不同的样式表时，这个属性值将会叠加几个样式表，遇到冲突的地方会以最后定义的为准。例如，首先链入一个外部样式表，其中定义了 h3 选择符的 color、text-align 和 font-size 属性。

```
h3 {color: red;
text-align: left;
font-size: 8pt
}
```

（标题 3 的文字颜色为红色，向左对齐，文字尺寸为 8 号字）

然后在内部样式表里也定义了 h3 选择符的 text-align 和 font-size 属性。

```
h3 {text-align: right;
font-size: 20pt
}
```

（标题 3 文字向右对齐，尺寸为 20 号字）

那么这个页面叠加后的样式就是：

```
color: red;
text-align: right;
font-size: 20pt
```

（文字颜色为红色，向右对齐，尺寸为 20 号字）

字体颜色从外部样式表里保留下来，而对齐方式和字体尺寸都有定义时，按照后定义

的优先而依照内部样式表。

注意：依照后定义的优先，优先级最高的是内嵌样式，内部样式表高于导入外部样式表，链入的外部样式表和内部样式表之间是最后定义的优先级高。

2.3　JavaScript 语言简介

2.3.1　JavaScript 语言概况

1. 什么是 JavaScript

JavaScript 是一种基于对象（Object）和事件驱动（Event Driven）并具有安全性能的脚本语言。使用它的目的是与 HTML、Java 脚本语言（Java 小程序）一起实现在一个 Web 页面中链接多个对象，与 Web 客户交互作用。从而可以开发客户端的应用程序等。它是通过嵌入或调入在标准的 HTML 语言中实现的。它的出现弥补了 HTML 语言的缺陷，也是 Java 与 HTML 折中的选择。它具有以下几个基本特点。

1）脚本语言。它采用小程序段的方式实现编程。像其他脚本语言一样，JavaScript 同样也是一种解释性语言，提供了一个易开发的过程。它的基本结构形式与 C、C ++、VB 和 Delphi 十分类似。但它不像这些语言那样需要先编译，而是在程序运行过程中被逐行地解释。它与 HTML 标识结合在一起，从而方便用户的使用操作。

2）基于对象的语言。JavaScript 是一种基于对象的语言，同时也可以看作一种面向对象的语言。这意味着它能运用自己已经创建的对象。因此，许多功能可以来自于脚本环境中对象的方法与脚本的相互作用。

3）简单性。JavaScript 的简单性主要体现在：首先，它是一种基于 Java 基本语句和控制流之上的简单而紧凑的设计，从而对于学习 Java 是一种非常好的过渡。其次，它的变量类型是采用弱类型，并未使用严格的数据类型。

4）安全性。JavaScript 是一种安全性语言，不允许访问本地的硬盘，并不能将数据存入到服务器上，不允许对网络文档进行修改和删除，只能通过浏览器实现信息浏览或动态交互，从而有效地防止数据的丢失。

5）动态性。JavaScript 是动态的，它可以直接对用户或客户输入做出响应，无须经过 Web 服务程序。它对用户的反应响应，是采用以事件驱动的方式进行的。事件驱动，就是指在主页（Home Page）中执行了某种操作所产生的动作，就称为“事件（Event）”。例如按下鼠标、移动窗口和选择菜单等都可以视为事件。当事件发生后，可能会引起相应的事件响应。

6）跨平台性。JavaScript 是依赖于浏览器本身，与操作环境无关，只要能运行浏览器的计算机，并支持 JavaScript 的浏览器就可正确执行。从而实现了“编写一次，走遍天下”的梦想。

实际上 JavaScript 最杰出之处在于可以用很小的程序做大量的事。无须高性能的计算机，仅需要一个文字处理软件及一浏览器，无须 Web 服务器通道，通过自己的计算机即

可完成所有的事情。

综上所述,JavaScript 是一种新的描述语言,它可以被嵌入到 HTML 的文件之中。JavaScript 语言可以做到回应使用者的需求事件(如 form 的输入),而不用任何的网络来回传输资料。所以当一位使用者输入一项资料时,它不用经过传给伺服端处理再传回来的过程,而直接可以被客户端的应用程式所处理。

2. 编写第一个 JavaScript 程序

下面通过一个例子,编写第一个 JavaScript 程序。通过它可说明 JavaScript 的脚本是怎样被嵌入到 HTML 文档中的。

```
<html>
<head>
<Script Language="JavaScript">
//JavaScript Appears here.
alert("这是第一个 JavaScript 例子!");
alert("欢迎你进入 JavaScript 世界!");
alert("今后我们将共同学习 JavaScript 知识!");
</Script>
</Head>
</Html>
```

在 Internet Explorer5.0 中运行后的结果如图 2-2 所示。

说明:test.html 是 HTML 文档,其标识格式为标准的 HTML 格式。如同 HTML 标识语言一样,JavaScript 程序代码是一些可用字处理软件浏览的文本,它在描述页面的 HTML 相关区域出现。JavaScript 代码由 <Script Language = " JavaScript " > … </Script>说明。在标识<Script Language="JavaScript">…</Script>之间就可加入 JavaScript 脚本。



图 2-2 程序运行的结果

alert()是 JavaScript 的窗口对象方法,其功能是弹出一个具有 OK 对话框并显示()中的字符串。通过<!——…//——>标识说明:若不认识 JavaScript 代码的浏览器,则所有在其中的标识均被忽略;若认识,则执行其结果。JavaScript 以</Script> 标签结束。

2.3.2 JavaScript 基本数据结构

JavaScript 提供脚本语言的编程与 C++非常相似,它只是去掉了 C 语言中有关指针等容易产生的错误,并提供了功能强大的类库。对于已经具备 C++或 C 语言的人来说,学习 JavaScript 脚本语言是一件非常轻松愉快的事。

1. JavaScript 代码的加入

JavaScript 的脚本包括在 HTML 中，它成为 HTML 文档的一部分。与 HTML 标识相结合，构成了一个功能强大的 Internet 网上编程语言。可以直接将 JavaScript 脚本加入文档：

```
<Script Language="JavaScript">
JavaScript 语言代码；
JavaScript 语言代码；
...
</Script>
```

说明：通过标识<Script>…</Script>指明 JavaScript 脚本源代码将放入其间。通过属性 Language="JavaScript"说明标识中是使用的何种语言，这里是 JavaScript 语言，表示在 JavaScript 中使用的语言。

下面是将 JavaScript 脚本加入 Web 文档中的例子。

```
<HTML>
<Head>
<Script Language="JavaScript">
document. Write("这是网络学校！");
document. close();
</Script>
</Head>
</HTML>
```

在浏览器的窗口中调用 test2. html，则显示“这是网络学校！”字串。

说明：Document. write()是文档对象的输出函数，其功能是将括号中的字符或变量值输出到窗口；document. close()是将输出关闭。可将<Script>…</Script>标识放入<Head>…</Head>或<Body>…</Body>之间。将 JavaScript 标识放入<Head>…</Head>之间，使之在主页和其余部分代码之前装载，从而可使代码的功能更强大。可以将 JavaScript 标识放置在<Body>…</Body>主体之间以实现某些部分动态地创建文档。

2. 基本数据类型

JavaScript 脚本语言同其他语言一样，有它自身的基本数据类型、表达式和算术运算符以及程序的基本框架结构。JavaScript 提供了 4 种基本的数据类型用来处理数字和文字，而变量提供存放信息的地方，表达式则可以完成较复杂的信息处理。

1) 基本数据类型

在 JavaScript 中有 4 种基本的数据类型：数值(整数和实数)、字符串型(用" "或' '号括起来的字符或数值)、布尔型(用 True 或 False 表示)和空值。在 JavaScript 的基本类型中，数据可以是常量，也可以是变量。由于 JavaScript 采用弱类型的形式，因而一个数

据的变量或常量不必首先作声明,而是在使用或赋值时确定其数据的类型。当然,也可以先声明该数据的类型,它是通过在赋值时自动说明其数据类型的。

2) 常量

- 整型常量:JavaScript 的常量通常又称字面常量,它是不能改变的数据。其整型常量可以使用十六进制、八进制和十进制表示其值。
- 实型常量:由整数部分加小数部分表示,如 12.32、193.98 等。可以使用科学或标准方法表示,如 5E7、4e5 等。
- 布尔值:布尔常量只有 True 和 False 两种状态。它主要用来说明或代表一种状态或标志,以说明操作流程。它与 C ++是不一样的,C ++可以用 1 或 0 表示其状态,而 JavaScript 只能用 True 或 False 表示其状态。
- 字符型常量:使用单引号(')或双引号(")括起来的一个或几个字符。如"This is a book of JavaScript"、"3245"和"ewrt234234" 等。
- 空值:JavaScript 中有一个空值 null,表示什么也没有。如试图引用没有定义的变量,则返回一个 Null 值。
- 特殊字符:同 C 语言一样,JavaScript 中同样有些以反斜杠(/)开头的不可显示的特殊字符。通常称为控制字符。

3) 变量

变量的主要作用是存取数据、提供存放信息的容器。对于变量必须明确变量的命名、变量的类型、变量的声明及其作用域。

(1) 变量的命名。

JavaScript 中的变量命名同其计算机语言非常相似,这里要注意以下两点。

- 必须是一个有效的变量,即变量以字母开头,中间可以出现数字如 test1、text2 等。除下划线(_)作为连字符外,变量名称不能有空格、"+"、"-"、","或其他符号。
- 不能使用 JavaScript 中的关键字作为变量。

在 JavaScript 中定义了 40 多个关键字,这些关键字是 JavaScript 内部使用的,不能作为变量的名称。如 Var、int、double 和 true 不能作为变量的名称。

在对变量命名时,最好把变量的意义与其代表的意思对应起来,以免出现错误。

(2) 变量的类型。

在 JavaScript 中,变量可以用命令 Var 作声明:Var mytest;。

该例子定义了一个 mytest 变量,但没有赋予它的值。

```
Var mytest="This is a book"
```

该例子定义了一个 mytest 变量,同时赋予了它的值。

在 JavaScript 中,变量可以不作声明,而在使用时再根据数据的类型来确定变量的类型。如:

```
x=100
y="125"
```

```
xy=True
cost=19.5
```

其中 x 为整数，y 为字符串，xy 为布尔型，cost 为实型。

(3) 变量的声明及其作用域。

JavaScript 变量可以在使用前先作声明，并可赋值。通过使用 var 关键字对变量作声明。对变量作声明的最大好处就是能及时发现代码中的错误，因为 JavaScript 是采用动态编译的，而动态编译是不易发现代码中的错误，特别是变量命名方面。

对于变量还有一个重要性——变量的作用域。在 JavaScript 中同样有全局变量和局部变量。全局变量是定义在所有函数体之外，其作用范围是整个函数；而局部变量是定义在函数体之内，只对该函数是可见的，而对其他函数则是不可见的。

3. 表达式和运算符

1) 表达式

在定义完变量后，就可以对它们进行赋值、改变和计算等一系列操作，这一过程通常由表达式完成。表达式是变量、常量、布尔及运算符的集合，因此表达式可以分为算术表达式、字串表达式、赋值表达式及布尔表达式等。

2) 运算符

运算符完成操作的一系列符号，在 JavaScript 中有算术运算符，如＋、－、＊、/等；有比较运算符，如!＝、＝＝等；有布尔逻辑运算符，如!(取反)、|、||等；有字串运算符，如＋、＋＝等。

在 JavaScript 中主要有双目运算符和单目运算符。其双目运算符由下列组成：

操作数 1 运算符 操作数 2

即由两个操作数和一个运算符组成，如 50＋40、"This"＋"that"等。单目运算符，只需一个操作数，其运算符可在前或后。

(1) 算术运算符。

JavaScript 中的算术运算符有单目运算符和双目运算符。

- 双目运算符：＋(加)、－(减)、＊(乘)、/(除)、%(取模)、|(按位或)、&(按位与)、<<(左移)、>>(右移)和>>>(右移，0 填充)。
- 单目运算符：－(取反)、~(取补)、＋＋(递加 1)和－－(递减 1)。

(2) 比较运算符。

比较运算符的基本操作过程是，首先对它的操作数进行比较，然后再返回一个 True 或 False 值。有如下 6 个比较运算符。

<(小于)、>(大于)、<＝(小于等于)、>＝(大于等于)、＝＝(等于)、!＝(不等于)

(3) 布尔逻辑运算符。

在 JavaScript 中增加了几个布尔逻辑运算符。

!(取反)、&＝(与之后赋值)、&(逻辑与)、|＝(或之后赋值)、|(逻辑或)、^＝(异或之后赋值)、^(逻辑异或)、?:(三目操作符)、||(或)、＝＝(等于)和!＝(不等于)

其中三目操作符主要格式如下：

操作数?结果 1：结果 2

若操作数的结果为真，则表达式的结果为结果 1，否则为结果 2。

3）范例

下面是一个跑马灯效果的 JavaScript 文档。

```
<html>
<head>
<script Language="JavaScript">
var msg="这是一个跑马灯效果的 JavaScript 文档";
var interval=100;
var spacelen=120;
var space10=" ";
var seq=0;
function Scroll() {
len=msg.length;
window.status=msg.substring(0, seq+1);
seq++;
if ( seq >=len ) {
seq=spacelen;
window.setTimeout("Scroll2();", interval );
}
else
window.setTimeout("Scroll();", interval );
}
function Scroll2() {
var out="";
for (i=1; i<=spacelen/space10.length; i++) out+=
space10;
out=out+msg;
len=out.length;
window.status=out.substring(seq, len);
seq++;
if ( seq >=len ) { seq=0; };
window.setTimeout("Scroll2();", interval );
}
Scroll();
</script>
<body>
</body>
</html>
```

上面介绍了 JavaScript 脚本是如何加入 Web 页面，并学习了 JavaScript 语言中的基本数据类型、变量、常量和操作运算符等。

2.3.3　JavaScript 程序构成

JavaScript 脚本语言的基本构成包括控制语句、函数、对象、方法和属性等，并由这些基本构成来实现编程。

1. 程序控制流

在任何一种语言中，程序控制流是必需的，它能使得整个程序减小混乱，使之顺利按其一定的方式执行。下面是 JavaScript 常用的程序控制流结构及语句。

1) if 条件语句

基本格式：

```
if(表达式)
语句段 1;
…
else
语句段 2;
…
```

功能：若表达式为 true，则执行语句段 1；否则执行语句段 2。

说明：

(1) if-else 语句是 JavaScript 中最基本的控制语句，通过它可以改变语句的执行顺序。

(2) 表达式中必须使用关系语句来实现判断，它是作为一个布尔值来估算的。

(3) 它将 0 和非 0 的数分别转化成 false 和 true。

(4) 若 if 后的语句有多行，则必须使用花括号将其括起来。

if 语句的嵌套：

```
if(布尔值)语句 1;
else(布尔值)语句 2;
else if(布尔值)语句 3;
…
else 语句 4;
```

在这种情况下，每一级的布尔表达式都会被计算，若为真，则执行其相应的语句，否则执行 else 后的语句。

2) For 循环语句

基本格式：

```
for(初始化;条件;增量)
语句集;
```

功能：实现条件循环，当条件成立时，执行语句集，否则跳出循环体。

说明：

(1) 初始化参数告诉循环的开始位置，必须赋予变量的初值。

(2) 条件：用于判别循环停止时的条件。若条件满足，则执行循环体，否则跳出。

(3) 增量：主要定义循环控制变量在每次循环时按什么方式变化。

(4) 三个主要语句之间，必须使用逗号分隔。

3) while 循环

基本格式：

```
while(条件)
语句集；
```

该语句与 For 语句一样，当条件为真时重复循环，否则退出循环。

For 与 while 语句都是循环语句，使用 For 语句在处理有关数字时更易看懂，也较紧凑；而 while 循环对复杂的语句效果更特别。

4) break 和 continue 语句

与 C ++语言相同，使用 break 语句使得循环从 For 或 while 中跳出，continue 使得跳过循环内剩余的语句而进入下一次循环。

2. 函数

函数为程序设计人员提供了一个非常方便的能力。通常在进行一个复杂的程序设计时，总是根据所要完成的功能，将程序划分为一些相对独立的部分，每部分编写一个函数。从而使各部分充分独立，任务单一，程序清晰，易懂、易读、易维护。JavaScript 函数可以封装那些在程序中可能要多次用到的模块，并可作为事件驱动的结果而调用的程序。从而实现一个函数把它与事件驱动相关联。这是与其他语言不一样的地方。

1) JavaScript 函数定义

```
Function 函数名 (参数,变元){
函数体；
Return 表达式；
}
```

说明：

当调用函数时，所用变量或字面量均可作为变元传递。函数由关键字 Function 定义。函数名，定义自己函数的名字。参数表，是传递给函数使用或操作的值，其值可以是常量、变量或其他表达式。通过指定函数名(实参)来调用一个函数，必须使用 Return 将值返回。函数名对大小写是敏感的。

2) 函数中的形式参数

在函数的定义中，我们看到函数名后有参数表，这些参数变量可能是一个或几个。那么怎样才能确定参数变量的个数呢？在 JavaScript 中可通过 arguments . Length 来检查参数的个数。例如：

```
Function function_Name(exp1,exp2,exp3,exp4)
Number=function _Name . arguments .length;
if (Number>1)
document.wrile(exp2);
if (Number>2)
document.write(exp3);
if(Number>3)
document.write(exp4);
...
```

3. 事件驱动及事件处理

1）基本概念

JavaScript 是基于对象(object-based)的语言。这与 Java 不同,Java 是面向对象的语言。而基于对象的基本特征,就是采用事件驱动(event-driven)。它是在图形界面的环境下,使得一切输入变得简单化。通常鼠标或热键的动作称为事件,而由鼠标或热键引发的一连串程序的动作,称为事件驱动。而对事件进行处理程序或函数,称为事件处理程序(Event Handler)。

2）事件处理程序

在 JavaScript 中对象事件的处理通常由函数(Function)担任。其基本格式与函数全部一样,可以将前面所介绍的所有函数作为事件处理程序。格式如下:

```
Function 事件处理名(参数表){
事件处理语句集;
:
}
```

3）事件驱动

JavaScript 事件驱动中的事件是通过鼠标或热键的动作引发的。它主要有以下几个事件。

(1) 单击事件 onClick。

当用户单击鼠标按钮时,产生 onClick 事件。同时 onClick 指定的事件处理程序或代码将被调用执行。通常在下列基本对象中产生。

- button(按钮对象)
- checkbox(复选框)或(检查列表框)
- radio (单选按钮)
- reset buttons(重要按钮)
- submit buttons(提交按钮)

例如,可通过下列按钮激活 change()文件。

```
<Form>
<Input type="button" Value=" " onClick="change()">
</Form>
```

在 onClick 等号后，可以使用自己编写的函数作为事件处理程序，也可以使用 JavaScript 中内部的函数。还可以直接使用 JavaScript 的代码等。例如：

```
<Input type="button" value=" " onclick=alert("这是一个例子");
```

(2) onChange 改变事件。

当利用 text 或 textarea 元素输入字符值改变时引发该事件，同时当在 select 表格项中一个选项状态改变后也会引发该事件。例如：

```
<Form>
<Input type="text" name="Test" value="Test" onCharge="check('this.test)">
</Form>
```

(3) 选中事件 onSelect。

当 text 或 textarea 对象中的文字被加亮后，引发该事件。

(4) 获得焦点事件 onFocus。

当用户单击 text 或 textarea 及 select 对象时产生该事件。此时该对象成为前台对象。

(5) 失去焦点 onBlur。

当 text 对象或 textarea 对象及 select 对象不再拥有焦点而退到后台时，引发该文件，它与 onFocus 事件是一个对应的关系。

(6) 载入文件 onLoad。

当文档载入时，产生该事件。onLoad 的一个作用就是在首次载入一个文档时检测 cookie 的值，并用一个变量为其赋值，使它可以被源代码使用。

(7) 卸载文件 onUnload。

当 Web 页面退出时引发 onUnload 事件，并可更新 Cookie 的状态。

4. 范例

范例 1：下例程序是一个自动装载和自动卸载的例子。即在装入 HTML 文档时调用 loadform()函数，而退出该文档进入另一个 HTML 文档时则首先调用 unloadform 函数，确认后方可进入。

```
<HTML>
<HEAD>
<script Language="JavaScript">
<!--
function loadform(){
```

```
alert("这是一个自动装载例子!");
}
function unloadform(){
alert("这是一个卸载例子!");
}
//-->
</Script>
</HEAD>
<BODY OnLoad="loadform()" OnUnload="unloadform()">
<a href="test.htm">调用</a>
</BODY>
</HTML>
```

范例 2：这是一个获取浏览器版本号的程序。该程序首先显示一个波浪，即提示信息。之后显示浏览器的版本号有关信息。

```
<html>
<head>
<script language="JavaScript"><!--
//-->
function makeArray(n){
this.length=n
return this
}
function hexfromdec(num) {
hex=new makeArray(1);
var hexstring="";
var shifthex=16;
var temp1=num;
for(x=1; x>=0; x--) {
hex[x]=Math.round(temp1/shifthex-.5);
hex[x-1]=temp1-hex[x] * shifthex;
temp1=hex[x-1];
shifthex /=16;
}
for (x=1; x>=0; x--) { hexstring+=getletter(hex[x]); }
return (hexstring);
}
function getletter(num) {
if (num<10) { return num; }
else {
if (num==10) { return "A" }
```

```
if (num==11) { return "B" }
if (num==12) { return "C" }
if (num==13) { return "D" }
if (num==14) { return "E" }
if (num==15) { return "F" }
}
}
function rainbow(text){
var color_d1;
var allstring="";
for(i=0;i<text.length;i=i+2){
color_d1=255*Math.sin(i/(text.length/3));
color_h1=hexfromdec(color_d1);
allstring+="<FONT COLOR="+color_h1+"ff"+color_h1+">"+text.substring(i,i+2)+
"</FONT>";
}
return allstring;
}
function sizefont(text){
var color_d1;
var allstring="";
var flag=0;
for(i=0,j=0;i<text.length;i=i+1){
if (flag==0) {
j++;
if (j>=7) {
flag=1;}}
if (flag==1) {
j=j-1;
if (j<=0) {
flag=0; }}
allstring+="<FONT SIZE="+j+">"+text.substring(i,i+1)+"</FONT>";
}
return allstring;
}
document.write("<font size=8><CENTER>")
document.write("<BR><BR>")
document.write( sizefont("这是一个获取 Web 浏览器的程序"))
document.write("</CENTER></font>")
document.write("浏览器名称："+navigator.appName+"<br>");
document.write("版本号："+navigator.appVersion+"<br>");
document.write("代码名字："+navigator.appCodeName+"<br>");
document.write("用户代理标识："+navigator.userAgent);
```

```
</script>
<body>
</body>
</html>
```

输出结果如图 2-3 所示。



图 2-3　输出结果

上面介绍了 JavaScript 程序设计的有关内容。程序流、函数和事件是学习掌握 JavaScript 编程的重点。

2.3.4　窗口及输入输出

JavaScript 是基于对象的脚本编程语言，其输入输出就是通过对象来完成的。其中有关输入可通过窗口对象来完成，而输出可通过文档对象的方法来实现。

1. 窗口及输入输出

请看下面例子：

```
<HTML>
<Head>
<script languaga="JavaScript">
Var test=window.prompt("请输入数据:");
document.write(test+"JavaScript 输入输出的例子");
</script>
</Head>
</HTML>
```

其中 window.prompt()是一个窗口对象的方法。其基本作用是当装入 Web 页面时在屏幕上显示一个具有“确定”和“取消”按钮的对话框，让输出数据。document.write 是一个文档对象的方法，它的基本功能是实现 Web 页面的输出显示，如图 2-4 所示。



图 2-4 Web 页的输出显示

1）窗口对象

该对象包括许多有用的属性、方法和事件驱动程序，编程人员可以利用这些对象控制浏览器，窗口显示的各个方面，如对话框、框架等。在使用时应注意以下几点。

该对象对应于 HTML 文档中的＜Body＞和＜FrameSet＞两种标识。

onload 和 onunload 都是窗口对象属性，在 JavaScript 脚本中可直接引用窗口对象。如 window.alert("窗口对象输入方法") 可直接使用以下格式：alert("窗口对象输入方法")。

2）窗口对象的事件驱动

窗口对象主要有装入 Web 文档事件 onload 和卸载时的 onunload 事件。用于文档载入和停止载入时开始和停止更新文档。

3）窗口对象的方法

窗口对象的方法主要用来提供信息或输入数据以及创建一个新的窗口。

(1) 创建一个新窗口。使用 window.open(参数表)方法可以创建一个新的窗口。其中参数表提供有窗口的主要特性和文档及窗口的命名。

(2) 创建具有 OK 按钮的对话框。alert 方法能创建一个具有 OK 按钮的对话框。

(3) 创建具有 OK 和 Cancel 按钮的对话框。confirm 方法为编程人员提供一个具有两个按钮的对话框。

(4) 创建具有输入信息的对话框。prompt 方法允许用户在对话框中输入信息，并可使用默认值，其基本格式如 prompt("提示信息",默认值)。

4）窗口对象中的属性

窗口对象中的属性主要用来对浏览器中存在的各种窗口和框架的引用，主要属性有以下几个。

(1) frames：确定文档中帧的数目。

frames(帧)作为实现一个窗口的分隔操作起到非常重要的作用，在使用时应注意，frames 属性是通过 HTML 标识＜Frames＞的顺序来引用的，它包含了一个窗口中的全部帧数。帧本身已是一类窗口，继承了窗口对象所有的属性和方法。

(2) Parent：指明当前窗口或帧的父窗口。

(3) defaultstatus：默认状态，它的值显示在窗口的状态栏中。

(4) status：包含文档窗口中帧中的当前信息。

(5) top：包括用以实现所有下级窗口的窗口。

(6) window：指的是当前窗口。

(7) self：引用当前窗口。

5）输出流及文档对象

在 JavaScript 文档对象中，提供了用于显示关闭、消除、打开 HTML 页面的输出流。

（1）创建新文档 open 方法。

使用 document.open 创建一个新的窗口或在指定的命令窗口内打开文档。由于窗口对象是所加载的父对象，因而在调用它的属性或方法时不需要加入 Window 对象。例如，用 Window. open 与 open 是一样的。打开一个窗口的基本格式：

```
Window .open("URL","窗口名称","窗口属性")
```

window 属性参数是一个字符串列表项，它由逗号分隔，它指明了有关新创建窗口的属性，如表 2-46 所示。

表 2-46　新创建窗口的属性参数

参　数	设　定　值	含　义
toolbar	yes/no	建立或不建立标准工具条
location	yes/no	建立或不建立位置输入字段
directions	yes/no	建立或不建立标准目录按钮
status	yes/no	建立或不建立状态条
menubar	yes/no	建立或不建立菜单条
scrollbar	yes/no	建立或不建立滚动条
revisable	yes/no	能否改变窗口大小
width	yes/no	确定窗口的宽度
height	yes/no	确定窗口的高度

在使用 open 方法时，需要注意以下几点。

① 通常浏览器窗口中，总有一个文档是打开的。因而不需要为输出建立一个新文档。

② 在完成对 Web 文档的写操作后，要使用或调用 close 方法来实现对输出流的关闭。

③ 在使用 open 来打开一个新流时，可为文档指定一个有效的文档类型，有效文档类型包括 text/HTML、text/gif、text/xim 和 text/plugin 等。

（2）write、writeln 输出显示。

该方法主要用来实现在 Web 页面上显示输出信息。在实际使用中，需注意以下几点。

- writeln 与 write 唯一不同之处在于在末尾加了一个换行符。
- 为了正常显示其输出信息，必须指明<Pre></Pre>标记，使之告诉编辑器。
- 输出的文档类型，可以由浏览器中有效的合法文本类型所确定。

（3）关闭文档流 close。

在实现多个文档对象中，必须使用 close 来关闭一个对象后，才能打开另一个文档

对象。

(4) 清除文档内容 clear。

使用该方法可清除已经打开的文档内容。

2. 简单的输入、输出例子

在 JavaScript 中可以非常方便地实现输入输出信息,并与用户进行交互。

1) JavaScript 信息的输入

通过使用 JavaScript 中所提供的窗口对象方法 prompt,就能完成信息的输入。该方法提供了最简便的信息输入方式,其基本格式如下:

```
Window.prompt("提示信息", 预定输入信息);
```

此方法首先在浏览器窗口中弹出一个对话框,让用户自行输入信息。一旦输入完成后,就返回用户所输入信息的值。例如:

```
test=prompt("请输入数据:","this is a JavaScript");
```

实际上 prompt 是窗口对象的一个方法。因为默认情况下所用的对象就是 window 对象,所以 windows 对象可以省略不写。

2) 输出显示

每种语言,都必须提供信息数据的输出显示。JavaScript 也一样,它提供了几个用于信息输出显示的方法。比较常用的有 window. alert、document. write 和 document. writln 方法。

(1) document. write 和 document. writeln 方法。

document 是 JavaScript 中的一个对象,在其中封装了许多有用的方法,其中 write 和 writeln 就是用于将文本信息直接输出到浏览器窗口中的方法。

```
document.write();document.writeln();
```

说明:write 和 writeln 方法都是用于向浏览器窗口输出文本字串;二者的唯一区别就是 writeln 方法自动在文本之后加入回车符。

(2) window. alert 输出。

在 JavaScript 中为了方便信息输出,JavaScript 提供了具有独立的对话框信息输出——alert 方法。alert 方法是 window 对象的一个方法,因此在使用时,不需要写 window 窗口对象名,而是直接使用就行了。它的主要用途在于输出时产生有关警告提示信息或提示用户,只有用户单击“确定”按钮后,方可继续执行其他脚本程序。例如:

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
<Script Language="JavaScript">
alert("这是一个 JavaScript 测试程序");
```

```
</Script>
</BODY>
</HTML>
```

(3) 利用输入、输出方法实现交互。

在 JavaScript 中,可以利用 prompt 和 write 方法实现与 Web 页面用户进行交互。下面就是一个有关实现交互的例子。

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
<Script Language="JavaScript">
<!--Hide From Other Browsers
document.write("<H1>有关交互的例子");
my=prompt("请输入数据:");
document.write(my+"</H1>");
document.close();
// Stop Hiding from Other Browsers-->
</Script>
</BODY>
</HTML>
```

从上面程序可以看出,可通过 write 和 prompt 方法实现交互。

在 JavaScript 脚本语言中,可以使用 HTML 标识语言的代码,从而实现混合编程。其中<H1>和
就是 HTML 标识符。

(4) 范例。

下列程序演示了进入主页所停留的时间。

```
<html>
<form name="myform">
<td vAlign="top" width="135">您在此停留了:
<input name="clock" size="8" value="在线时间"></td>
</form>
<script language="JavaScript">
var id, iM=0, iS=1;
start=new Date();
function go()
{ now=new Date();
time=(now.getTime()-start.getTime()) / 1000;
time=Math.floor(time);
iS=time % 60;
```

```
iM=Math.floor(time/60);
if (iS<10)
document.myform.clock.value=" "+iM+" 分 0"+iS+" 秒";
else
document.myform.clock.value=" "+iM+" 分 "+iS+" 秒";
id=setTimeout("go()", 1000);
}
go();
</script>
</body>
</html>
```

在浏览器中的结果如图 2-5 所示。



图 2-5 程序运行结果

2.3.5 Web 页面信息的交互

要实现动态交互，必须掌握有关窗体对象（Form）和框架对象（Frames）更为复杂的知识。

1. 窗体基础知识

窗体对象可以使设计人员用窗体中不同的元素与客户机用户相交互，而用不着在之前首先进行数据输入，就可以实现动态改变 Web 文档的行为。

1）什么是窗体对象

窗体构成了 Web 页面的基本元素。通常 Web 页面有一个窗体或几个窗体，使用 Forms[]数组来实现不同窗体的访问。

```
<form Name=Form1>
<INPUT type=text…>
<Input type=text…>
<Inpup type=text…>
</form>
<form Name=Form2>
```

```
<INPUT type=text…>
<Input type=text…>
</form>
```

在 Forms[0]中共有三个基本元素，而 Forms[1]中只有两个元素。

窗体对象最主要的功能就是能够直接访问 HTML 文档中的窗体，它封装了相关的 HTML 代码。

```
<Form>
Name="表的名称"
Target="指定信息的提交窗口"
action="接收窗体程序对应的 URL"
Method=信息数据传送方式(get/post)
enctype="窗体编码方式"
[onsubmit="JavaScript 代码"]>
</Form>
```

2）窗体对象的方法

窗体对象的方法只有一个——submit 方法，该方法的主要功能是实现窗体信息的提交。如提交 Mytest 窗体，则使用下列格式：

```
document.mytest.submit()
```

3）窗体对象的属性

窗体对象的属性主要包括 elements、name、action、target、encoding 和 method。

除 elements 外，其他几个均反映了窗体的标识中相应属性的状态，这通常是单个窗体标识；而 elements 常常是多个窗体元素值的数组。例如：

```
elements[0].Mytable.elements[1]
```

4）访问窗体对象

在 JavaScript 中访问窗体对象可由以下两种方法实现。

（1）通过访问窗体。

在窗体对象的属性中首先必须指定其窗体名，而后就可以通过下列标识访问窗体，如 document. Mytable()。

（2）通过数组来访问窗体。

除了使用窗体名来访问窗体外，还可以使用窗体对象数组来访问窗体对象。但需要注意一点，因窗体对象是由浏览器的环境提供的，而浏览器环境所提供的数组下标是 0～n，所以可通过下列格式实现窗体对象的访问：

```
document.forms[0];document.forms[1];document.forms[2]…
```

5）引用窗体的先决条件

在 JavaScript 中要对窗体引用的条件是，必须先在页面中用标识创建窗体，并将定义

窗体部分放在引用之前。

2. 窗体中的基本元素

窗体中的基本元素由按钮、单选按钮、复选按钮、提交按钮、重置按钮和文本框等组成。在 JavaScript 中要访问这些基本元素，必须通过对应特定的窗体元素的数组下标或窗体元素名来实现。每一个元素主要是通过该元素的属性或方法来引用。其引用的基本格式如下：

```
formName.elements[].methadName (窗体名.元素名或数组.方法)
formName.element[].propertyName(窗体名.元素名或数组.属性)
```

下面分别介绍。

1）Text(单行单列输入元素)

功能：对 Text 标识中的元素实施有效的控制。

属性。

- Name：设定提交信息时的信息名称。对应于 HTML 文档中的 Name。
- Value：用以设定出现在窗口中对应 HTML 文档中 Value 的信息。
- defaultvalue：包括 Text 元素的默认值。

方法

blur：将当前焦点移到后台。

select：加亮文字。

事件。

onFocus：当 Text 获得焦点时，产生该事件。

OnBlur：从元素失去焦点时，产生该事件。

Onselect：当文字被加亮显示后，产生该文件。

onchange：当 Text 元素值改变时，产生该文件。

例如：

```
<Form name="test">
<input type="text" name="test" value="this is a javascript" >
</form>
...
<script language="Javascirpt">
document.mytest.value="that is a Javascript";
document.mytest.select();
document.mytest.blur();
</script>
```

2）textarea(多行多列输入元素)

功能：实施对 Textarea 中的元素进行控制。

属性。

- Name：设定提交信息时的信息名称，对应 HTML 文档 Textarea 的 Name。

- Value：用以设定出现在窗口中对应 HTML 文档中 Value 的信息。
- Default value：元素的默认值。

方法。

blur：将失去输入焦点。

select：将文字加亮。

事件。

- onBlur：当失去输入焦点后产生该事件。
- onFocus：当输入获得焦点后，产生该文件。
- Onchange：当文字值改变时，产生该事件。
- Onselect：当文字加亮后，产生该文件。

3）Select（选择元素）

功能：实施对滚动选择元素的控制。

属性。

- Name：设定提交信息时的信息名称，对应文档 select 中的 name。
- Length：对应文档 select 中的 length。
- Options：组成多个选项的数组。
- selectIndex：该下标指明一个选项。

在 select 中每一选项都含有以下属性。

- Text：选项对应的文字。
- selected：指明当前选项是否被选中。
- Index：指明当前选项的位置。
- defaultselected：默认选项。

事件。

- OnBlur：当 select 选项失去焦点时，产生该文件。
- OnFocus：当 select 获得焦点时，产生该文件。
- Onchange：选项状态改变后，产生该事件。

4）Button（按钮）

功能：实施对按钮的控制。

属性。

- Name：设定提交信息时的信息名称，对应文档中 button 的 Name。
- Value：用以设定出现在窗口中对应 HTML 文档中 Value 的信息。

方法。

click：类似于一个按下的按钮。

事件。

onclick：当单击按钮时，产生该事件。

例如：

```
<Form name="test">
<input type="button" name="testcall" onclick=tmyest()>
```

```
</form>
...
<script language="javascirpt">
document.elements[0].value="mytest";          //通过元素访问
document.testcallvalue="mytest";              //或通过名字访问
</script>
...
```

5）checkbox（检查框）

功能：实施对一个具有复选框中元素的控制。

属性。

- Name：设定提交信息时的信息名称。
- Value：用以设定出现在窗口中对应 HTML 文档中 Value 的信息。
- Checked：该属性指明框的状态 true/false。
- defaultchecked：默认状态。

方法。

click：使得框的某一个项被选中。

事件。

onclick：方框的选项被选中时，产生该事件。

6）radio（单选按钮）

功能：实施对一个具有单选功能的无线按钮控制。

属性。

- name：设定提交信息时的信息名称，对应 HTML 文档中 radio 的 name。
- value：用以设定出现在窗口中对应 HTML 文档中 Value 的信息，对应 HTML 文档中 radio 的 name。
- length：单选按钮中的按钮数目。
- defaultchecked：默认按钮。
- checked：指明选中还是没有选中。
- index：选中按钮的位置。

方法。

chick：选定一个按钮。

事件。

onclick：单击按钮时，产生该事件。

7）hidden（隐藏）

功能：实施对一个具有不显示文字并能输入字符的区域元素的控制。

属性。

- Name：设定提交信息时的信息名称，对应 HTML 文档的 hidden 中的 Name。
- Value：用以设定出现在窗口中对应 HTML 文档中 Value 的信息，对应 HTML 文档中 hidden 的 value。

• defaultvalue：默认值。

8）Password（密码）

功能：实施对具有密码输入的元素的控制。

属性。

• Name：设定提交信息时的信息名称，对应 HTML 文档中 password 的 name。
• Value：用以设定出现在窗口中对应 HTML 文档中 Value 的信息，对应 HTML 文档中 password 的 Value。
• defaultvalue：默认值。

方法。

• select：加亮输入密码域。
• blur：丢失 password 输入焦点。
• focus：获得 password 输入焦点。

9）submit（提交元素）

功能：实施对一个具有提交功能按钮的控制。

属性。

Name：设定提交信息时的信息名称，对应 HTML 文档中的 submit。

Value：用以设定出现在窗口中对应 HTML 文档中 Value 的信息，对应 HTML 文档中的 value。

方法。

click：相当于按下 submit 按钮。

事件。

onclick：当按下该按钮时，产生该事件。

3. 范例

下面演示通过单击一个按钮（red）来改变窗口颜色，单击“调用动态按钮文档”（如图 2-6）调用一个动态按钮文档，该动态按钮文档如图 2-7 所示。

```
<html>
<head>
<Script Language="JavaScript">
document.bgColor="blue";
document.vlinkColor="white";
document.linkColor="yellow";
document.alinkcolor="red";
//动态改变颜色
function changecolor(){
   document.bgColor="red";
   document.vlinkColor="blue";
   document.linkColor="green";
   document.alinkcolor="blue";
```

```
}
</script>
</HEAD>
<body bgColor="White" >
<A href="test8_2.htm">调用动态按钮文档</a>
<form >
   <Input type="button" Value="red" onClick="changecolor()">
</form>
</BODY>
</HTML>
```



图 2-6　动态按钮文档

```
<HTML>
<HEAD>
</HEAD>
<p align="center"></p>
<div align="center"><center>
<table border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td width="100%  "><form name="form2" onSubmit="null">
     <p><input type="submit" name="banner" VALUE="Submit"
     onClick="alert('You have to put an \'action=[url]\' on the form tag!!')"><br>
<script language="JavaScript">
var id,pause=0,position=0;
function banner() {
         // variables declaration
        var i,k,msg="这里输入你要的内容";// increase msg
         k=(30/msg.length)+1;
         for(i=0;i<=k;i++) msg+=" "+msg;
         // show it to the window
        document.form2.banner.value=msg.substring(position,position-30);
         // set new position
```

```
        if(position++==msg.length) position=0;
         // repeat at entered speed
         id=setTimeout("banner()",60); }
// end-->
banner();
</script></p>
   </form>
   </td>
  </tr>
</table>
</center></div>
<p>  </p>
<BODY>
<A href="test8_1.htm">返回</a>
</BODY>
</HTML>
```



图 2-7　程序运行结果

本章介绍了使用 JavaScript 脚本实现 Web 页面信息交互的方法，其中主要介绍了窗体中基本元素的主要功能和使用。

第 3 章　ASP.NET 基础

本章主要讲述 ASP.NET 的基础知识，包括开发环境的配置、C# 语言基础、ASP.NET 的一些常用控件及 ASP.NET 的基本对象。通过本章学习将会对 ASP.NET 有一定的了解，为以后开发系统奠定基础。

3.1　Visual Studio.NET 2005

Visual Studio.NET 是.NET 的最佳开发工具。Visual Studio.NET 是一套完整的开发工具，用于生成 ASP.NET Web 应用程序、XML Web Services、桌面应用程序和移动应用程序。Visual Basic.NET、Visual C++.NET 、Visual C#.NET 和 Visual J#.NET 全部使用相同的集成开发环境(Integrated Development Environment，IDE)，该环境允许它们共享工具并有助于创建混合语言解决方案。另外，这些语言利用了.NET Framework 的功能，此框架提供对简化 ASP.NET Web 应用程序和 XML Web Services 开发关键技术的访问。

3.1.1　导入导出设置

Visual Studio.NET 2005 包含许多对开发环境新的增强、创新和提高。图 3-1 为启动 Visual Studio.NET 2005 的常规界面。在 Visual Studio.NET 2005 发布以前，IDE 自定义的主要不足之处是，无法以可移植的格式保存首选项。使用 Visual Studio.NET 2005，可以在另一台计算机上快速地还原个人设置。Visual Studio.NET 2005 允许以如下方式使用自定义设置：制作当前设置的副本以供另一台计算机使用；将设置分发给其他开发人员；允许团队对 IDE 的某些元素使用相同的设置，而在 IDE 的其他区域内保留个人的自定义设置。

选择“工具”→“选项”命令，在“选项”对话框的“环境”结点中，选择“导入和导出设置”项，如图 3-2 所示。在“将我的设置自动保存到此文件”文本框中，指定的文件在每次关闭 Visual Studio 时都会进行更新。它可以是本地文件，也可以是网络文件。在经常操作两台计算机的情况下，将该文件设置在两台计算机都可以访问的网络位置，这样能够确保在两台计算机上均享用相同的 Visual Studio“外观”。另外，每次更新其中一台计算机设置时，它都会自动在另一台计算机上显示。



图 3-1　Visual Studio. NET 2005 启动后的初始页

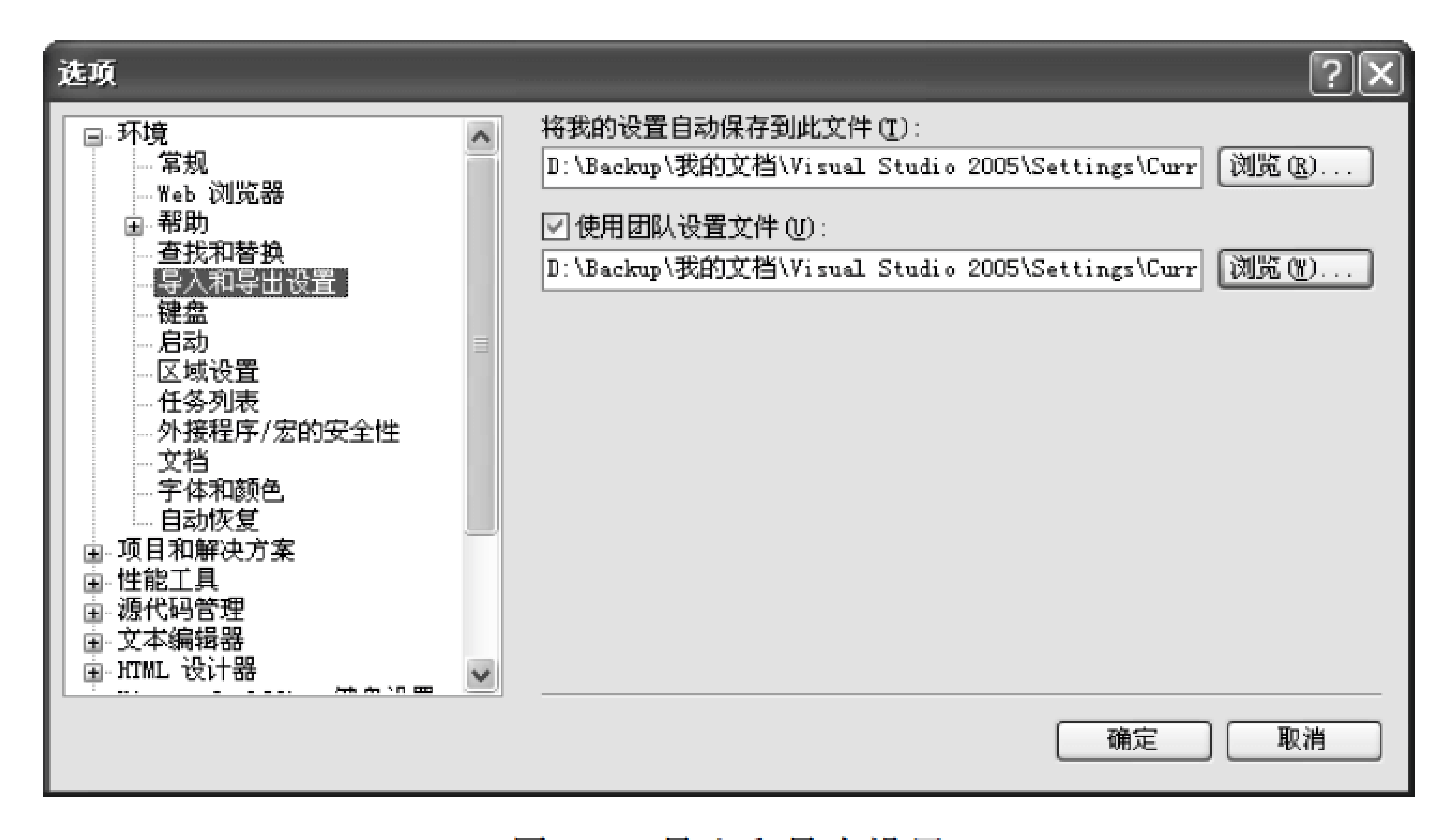


图 3-2　导入和导出设置

在“使用团队设置文件”复选框下的文本框中，可指定一个. vssettings 文件夹包含在一组开发人员中的共享设置。一个开发团队的所有成员都必须使用 Visual Studio. NET 代码，文件应该具有相同的缩进和格式化选项。开发人员主管可以配置 Visual Studio . NET 2005 中格式化选项，然后使用导入导出设置对话框将这些格式化设置保存到网络共享. vssettings 文件中。然后，开发团队的其他成员更新他们的 Visual Studio 配置以使用该团队的设置文件。如果该团队以后决定更新默认值，则开发人员主管可以将新设置导出到相同的文件位置，该团队中每个成员在他们下次启动 Visual Studio 时都将自动接收到此新设置。

决定更新默认值，则开发人员主管可以将新设置导出到相同的文件位置，该团队中每

个成员在他们下次启动 Visual Studio 时都将自动接收到此新设置。

首次启动 Visual Studio. NET 2005 时所选的设置，包括已安装. vssettings 文件中的预定义、自定义，以及其后进行的任何 IDE 的自定义，在每次退出 Visual Studio 时，活动设置都会自动保存到 currentsettings. vssettings 文件中。

当用户希望改变或者重设置工作环境时，可以选择“工具”→“导入和导出设置”命令。在弹出的对话框中，如图 3-3 所示，有如下三种选择：导出选定的环境设置、导入选定的环境设置和重置所有设置。该对话框可以让用户分别选择将那些设置导出到. vssettings 文件，或者从现有的. vssettings 文件导入那些设置，如图 3-4 所示。

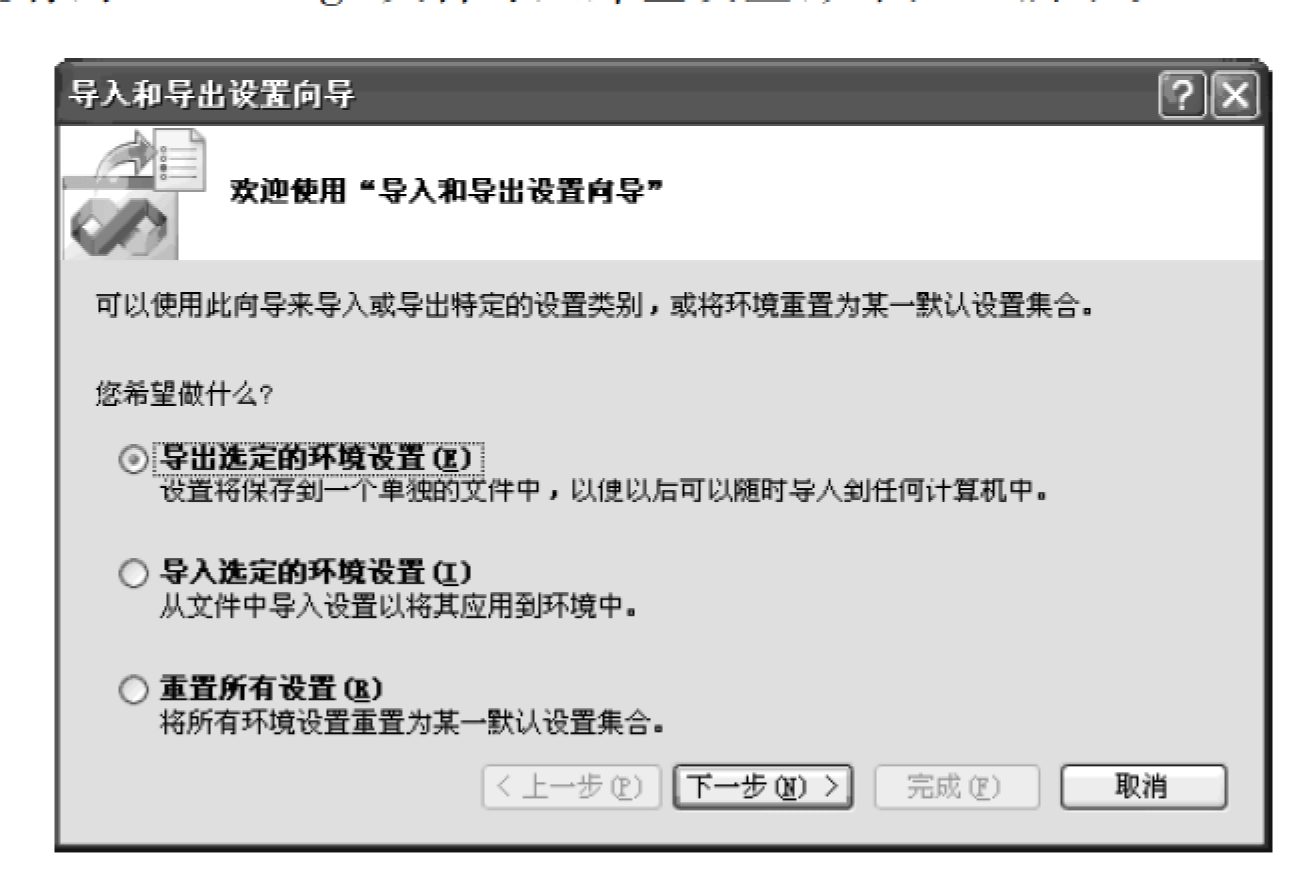


图 3-3 “导入和导出设置向导”对话框

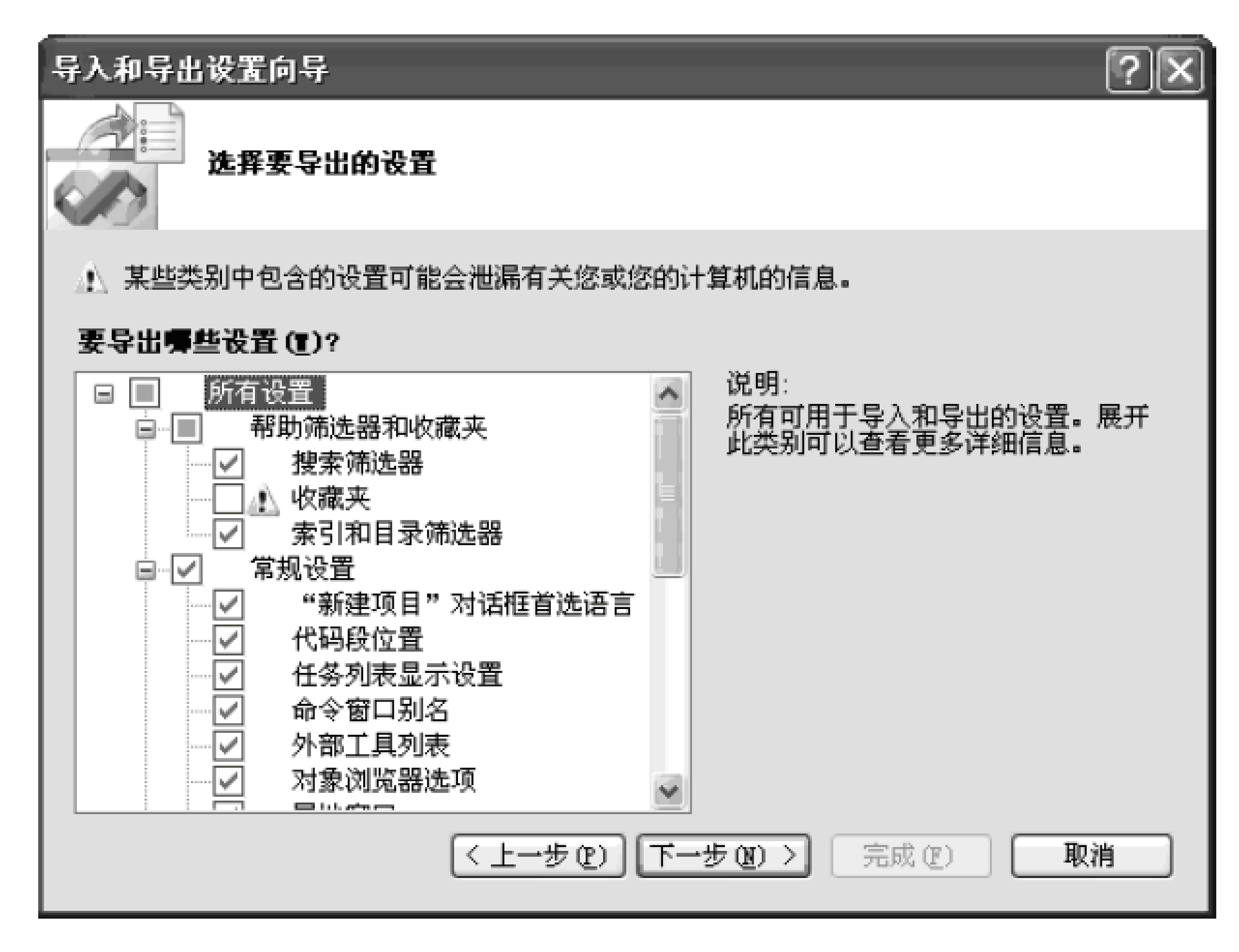


图 3-4 选择导入导出的设置

通过 Visual Studio. NET 2005 的“导入和导出设置”，可以灵活地控制 IDE 开发的设置，这对个人开发和团队开发都非常便利、快捷。

3.1.2　常用窗口

1. “工具箱”窗口

与 Visual Studio.NET 2003 相比，Visual Studio.NET 2005 工具箱的一个重要的变化是包含更加丰富的工具控件，图 3-5 所示为“工具箱”窗口。



图 3-5　“工具箱”窗口

根据所创建应用程序的类型，工具箱中将包含不同的控件群。以编写 ASP.NET 应用程序为例，默认情况下，打开一个.aspx 文件后，工具箱将出现 8 个控件群。每个控件群中包含多个功能相似的控件，下面是对 8 个控件群的简要说明。

（1）标准：包含最常用的控件。所有控件有一个共同特征，代码中都包含＜asp:＞。例如，常见的 Textbox、FileUpload 和 Wizard 等。

（2）数据：包含多个数据源和数据绑定控件，这些控件能够连接和显示不同数据源数据，如 SQL Server、Access 和 XML 等。

（3）验证：包含所有与数据验证有关的控件，如 RequiredFieldValidator、RangeValidator 等。利用这些控件可快速实现数据验证。

（4）导航：用于实现站点导航功能，如 Menu、SiteMapPath 和 TreeView 等。

（5）登录：包含与用户登录有关的服务器控件，如 Login、LoginView、LoginName 和 CreateUserWizard 等。

（6）WebParts：包含所有 Web 部件控件，如 WebPartZone、WebPartManager 等。

（7）HTML：包含常用 HTML 控件，如 Button、Table 等。

（8）常规：不包含任何控件。可以将自定义的常用控件添加到该控件群中。

2. “属性”窗口

“属性”窗口主要用于显示选定对象的具体属性信息，利用“属性”窗口可方便修改对象属性。图 3-6 所示的是按钮控件的相关属性。



图 3-6　“属性”窗口

“属性”窗口上部有一个下拉列表框，框中显示了当前被选中对象的名称和完整类名。可以利用该下拉列表框对页面中的多个对象进行选择，进而查看属性信息。在下拉列表框的下部是一个工具条，工具条中包含按分类顺序查看属性、按字母顺序查看其属性、事件和查看属性页的功能按钮。“属性”窗口中部是信息列表。所有与选定对象相关的属性或者事件信息等都显示在这个区域。“属性”窗口的下部是说明区域。由于属性众多，用户不可能对每个属性都了如指掌。因此，说明区域的设置则起到了很大的辅助作用。在属性信息列表中，单击任何

属性时，信息说明区域将立刻显示属性的简单说明。

3. 解决方案资源管理器

“解决方案资源管理器”是 Visual Studio 2005 中最为重要的功能窗口，它可以完成对解决方案中代码文件、图片等资源的管理任务。例如，对资源的复制、粘贴和删除等。另外，还能够实现对解决方案的配置和操作等。例如，编译项目、查看属性、添加文件和项设置启动项目等。图 3-7 所示为“解决方案资源管理器”窗口的截图。

“解决方案资源管理器”窗口中包含一个存储在“E:\项目\EnglishView\”中的 Web 站点项目。该项目中文件以树形结构显示在窗口中，同时，在窗口上部还包含一个工具条。图 3-8 所示为这些按钮的说明。

在“解决方案资源管理器”窗口的上部设置了一个工具条。该工具条中包含 7 个常用的功能按钮，这些按钮包含属性、刷新、嵌套相关文件、查看代码、查看设计器、复制网站和 ASP. NET 配置。用户可根据需要选择使用这些按钮。另外，工具条中的按钮也会随着不同的选择对象呈现动态变化。例如，图 3-7 所示的是. aspx 文件后的工具条状态。如果选中项目文件夹，则工具条还会显示“查看类关系图”按钮，并隐藏“查看代码”和“查看设计器”按钮。

4. 服务器资源管理器

默认情况下，“服务器资源管理器”窗口与“解决方案资源管理器”窗口处于同一位置，二者采用类似选项卡的形式重叠放置在一起。通过单击选项卡，可以在两个窗口之间进行切换。图 3-9 为“服务器资源管理器”窗口的截图。

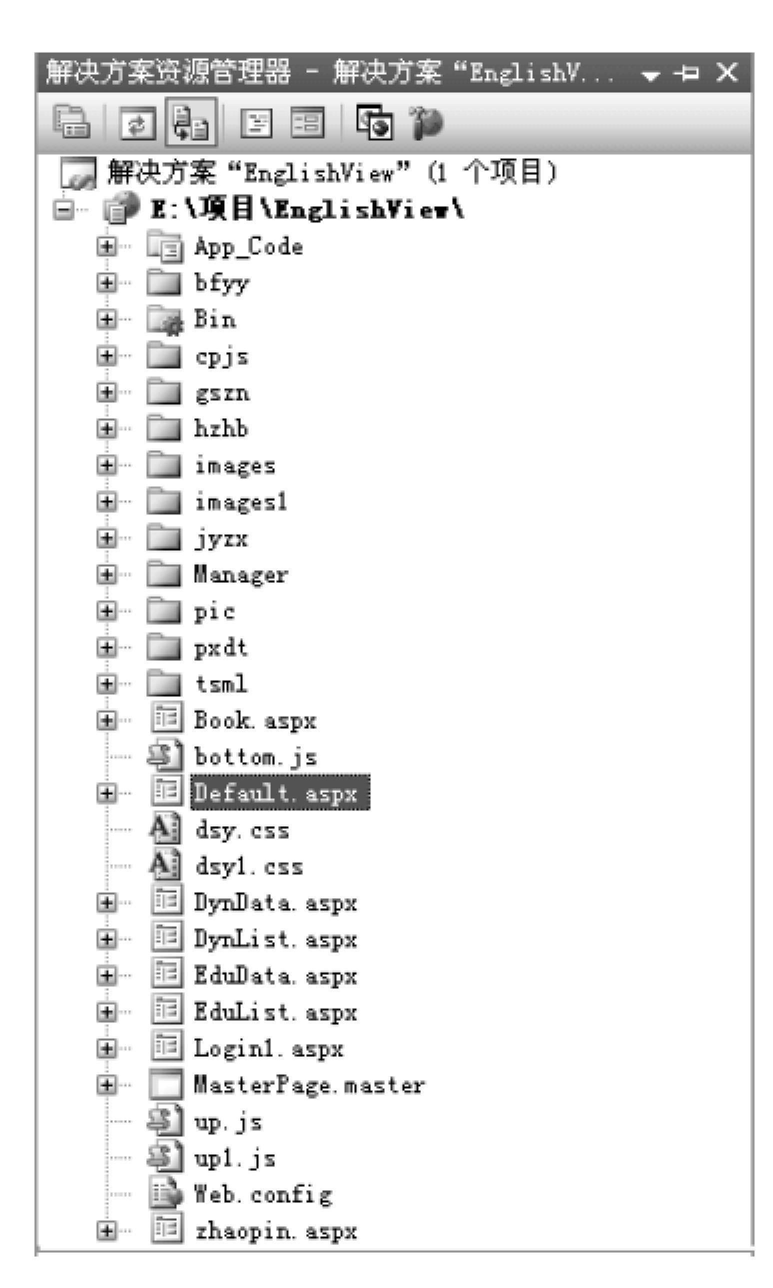


图 3-7 “解决方案资源管理器”窗口

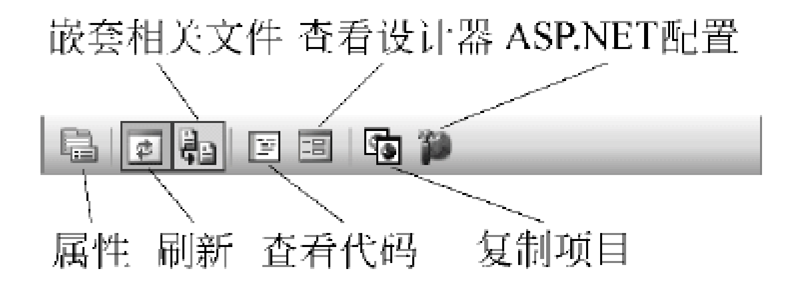


图 3-8 解决方案资源管理器中的工具条

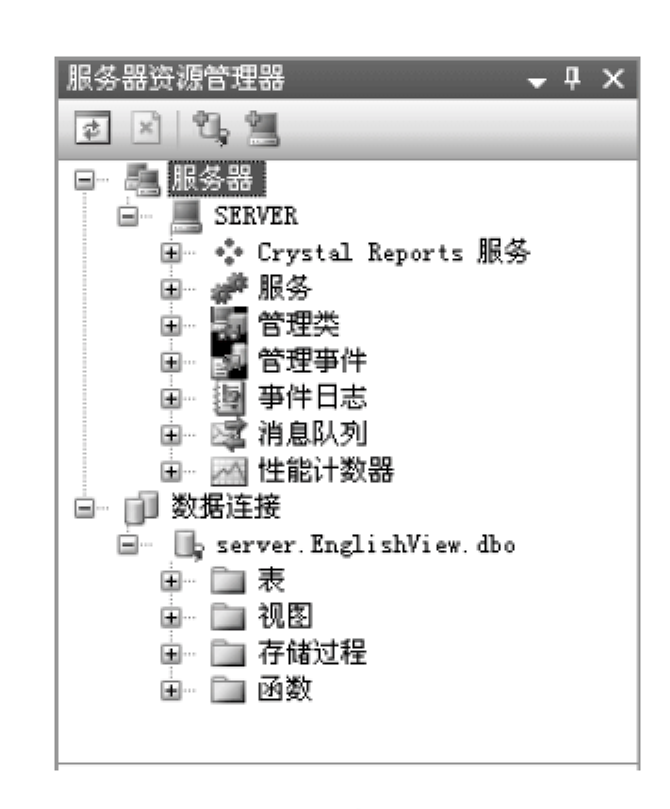


图 3-9 “服务器资源管理器”窗口

“服务器资源管理器”窗口有一个资源树,树中包含数据连接、服务和管理类等内容。利用这个服务器资源树,能够方便地完成如连接数据库、连接服务器和查看服务信息等任务。

5. 类视图

在类视图中,如图 3-10 所示,用户可以清晰地得到当前项目类的组织情况,以及每个类的基类和接口、成员变量、函数等。双击相应的变量、函数,则会直接定位到其相应代码处。在类视图对应工具栏中,用户可以改变类视图的排序方式,也可以建立新的文件夹。同样,用户需要注意这里丰富的右键菜单。而且,右键指向的位置不同(指向子结点还是根结点),其对应的右键菜单也会不同,就可以带来不同的快捷操作。

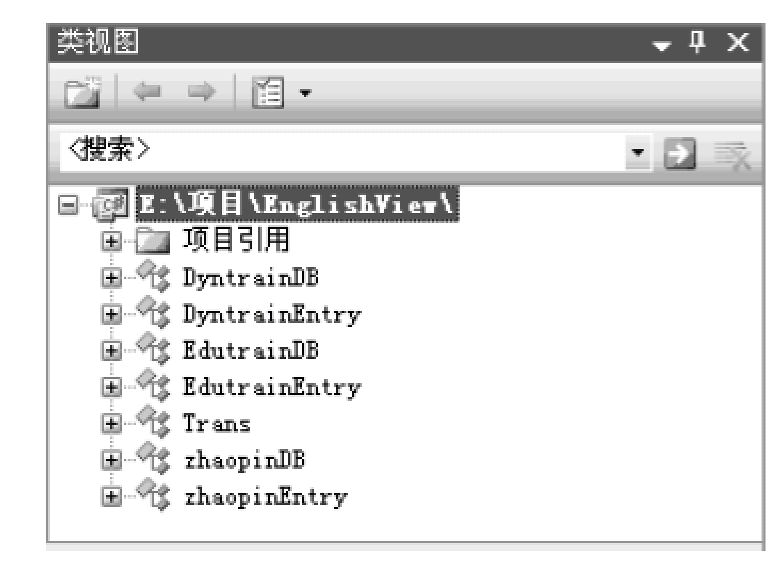


图 3-10　类视图

同时,类视图还是一种可视化设计工具,用于检查和操作类和其他类型的结构。该工具与设计器所显示的类型底层的源代码完全集成,例如,如果更改了一个属性名,该代码文件会自动进行更新以反映此更改。反过来,在源代码中所做的更改也会立即影响到设计器中与相关联对象的外观。

3.1.3　创建 Welcome Web 应用程序

1. 启动 Visual Studio 2005 建立 Web 网站

当第一次启动 Visual Studio 2005 时,系统会自动进行相关的配置。选择“开始”→“所有程序”→Microsoft Visual Studio 2005 命令,启动 Visual Studio 2005。

Visual Studio 2005 可以建立 HTTP 或文件系统网站。文件系统网站是保存在硬盘文件夹的 Web 网站,需要发布至 Web 服务器,或使用内建服务器测试 ASP. NET 程序的执行,其步骤如下。

(1) 启动 Visual Studio 2005,然后选择“文件”→“新建”→“网站”命令,打开“新建网站”对话框,如图 3-11 所示。



图 3-11　“新建网站”对话框

(2) 在“模板”列表框中选择“ASP. NET 网站”，在“位置”下拉列表中选择“文件系统”选项，就可以选择文件夹来建立文件系统网站。本例选择“文件系统”选项，在其后的文本框中输入 F：\Welcome，单击“确定”按钮创建网站。

(3) 网站默认建立 Default. aspx 网站首页文件和 Default. aspx. cs，并且建立 App_Data 子目录，如图 3-12 所示。

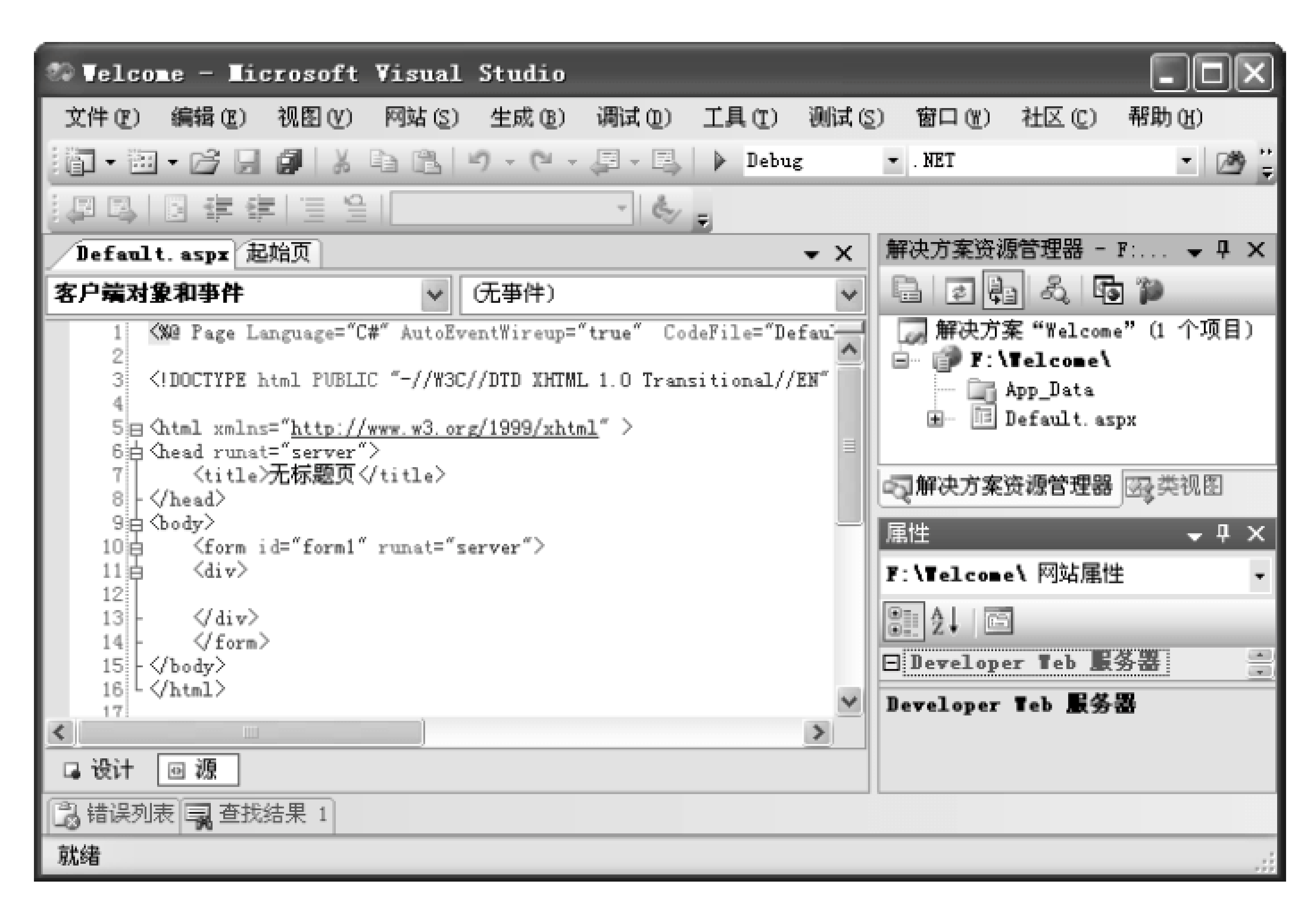


图 3-12　新建的网站

2. 添加内容

通过 Visual Studio 2005 提供的工具箱可以向页面添加各种控件，本例简单地向页面中添加一个 Label 控件来显示 Welcome 消息，具体步骤如下所示。

(1) 单击图 3-13 所示窗口底部的“设计”按钮，进入“设计”窗口。

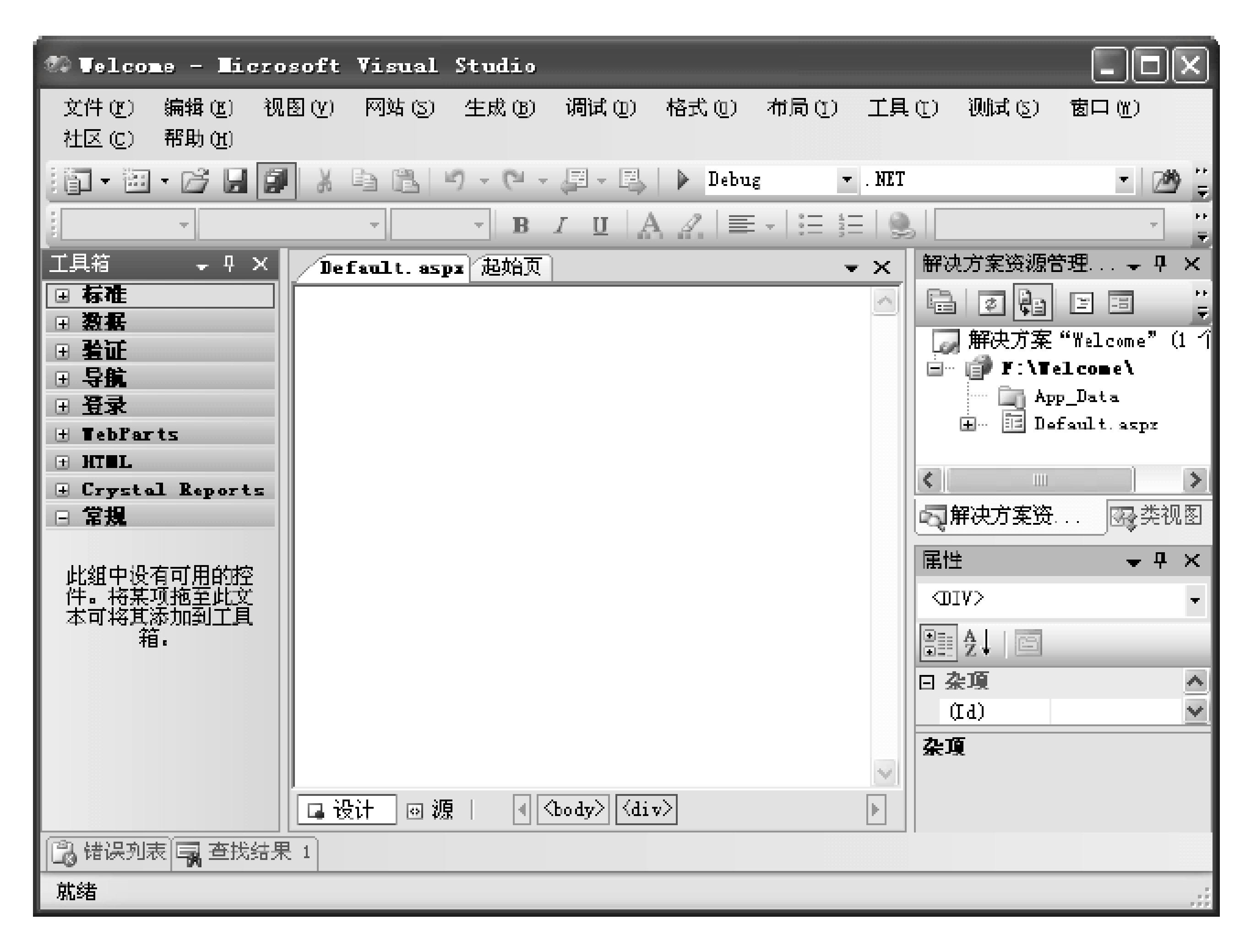


图 3-13　“设计”窗口

（2）将 Lable 控件拖入“设计”窗口，定义该控件的 ID、Text、Font 和 ForeColor 属性，如图 3-14 所示。

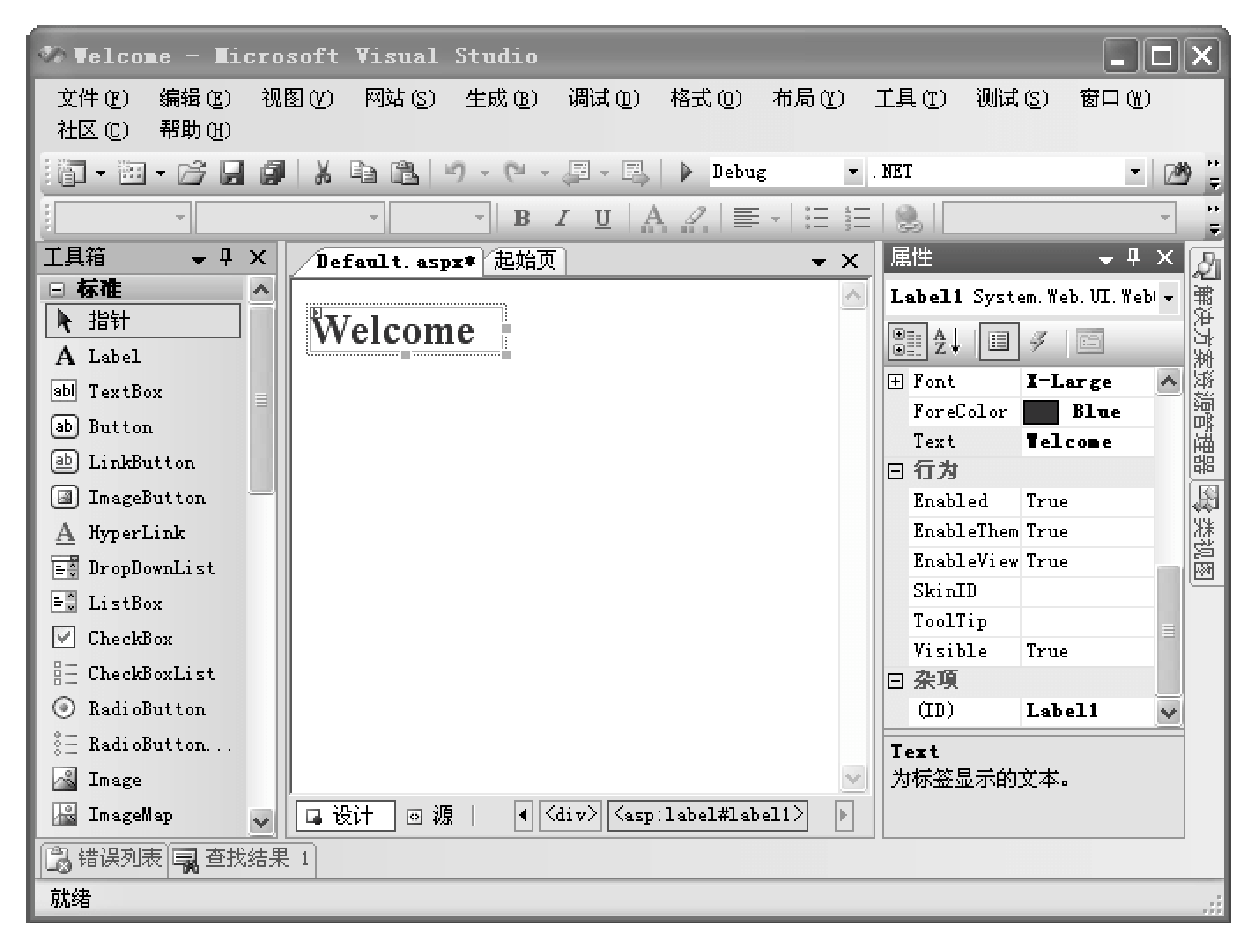


图 3-14　设置控件

（3）单击工具栏上的“保存”按钮，保存文件。

3. 调试 ASP.NET 程序

选择“调试”→“启动调试”命令，即可在浏览器中查看该程序，如图 3-15 所示。



图 3-15　Welcome 页面

这样，ASP.NET 网站的整个实现过程就演示完毕了。

3.2　C#基础

3.2.1　C#概述

1. C#和.NET 框架

1）什么是.NET 框架

.NET 定义了一种支持高度分散、基于组件的应用程序的开发和执行环境。它允许

不同计算机语言一起工作，并给 Windows 平台提供安全性、程序可移植性和通用程序模型。

2) C# 和.NET 框架的关系

C# 和运行环境.NET 框架具有特殊关系。

(1) Microsoft 最初设计 C# 是为了给.NET 框架创建代码。

(2) C# 使用的库就是.NET 框架所定义的库。

由于.NET 框架与 C# 相关，为了让多种不同的计算机语言在一起工作，.NET 定义了两个重要的实体。

(1) 通用语言运行时(Common Language Runtime, CLR)，这是管理程序执行的系统。CLR 是.NET 框架中允许程序移植、支持混合语言设计和提供安全性的一部分。

(2) .NET 类库，该库让程序有权访问运行环境。只要程序满足.NET 类库定义的特征，它就能在任何支持.NET 的地方运行。

2. C# 程序实例

现在来看一个实际的 C# 程序，编译并运行程序 3-1。

程序 3-1　实现简单的输出

```
using System;
class Example
{
  Public static void Main ( )
  {
    Console.WriteLine ("A simple C#  program.");
  }
}
```

使用 Visual Studio 集成开发环境编辑、编译和运行 C# 程序。下面是关于 C# 程序的说明。

(1) C# 程序的名称是任意的，只要遵循其扩展名为.CS 即可。例如本实例程序名为 Example.cs。

(2) C# 中大写和小写字母是有区别的。例如不小心将 Main 输入成 main，或者将 WriteLine 输入成 writeline，那么上面的程序将是错误的。

(3) C# 使用如下三种注释方法。

多行注释：/ * … * /

单行注释：//

XML 注释：///

(4) 语句 using System; 表示程序使用 System 名字空间。在 C# 中名字空间(namespace)定义声明范围。名字空间提供使一组名称与其他名称分开的方式。基本上，一个名字空间中声明的名称将不会与另一个名字空间中声明的名称相冲突。如果需要，可以用它所属名字空间完全限定名称。

(5) class Example 语句,使用关键字 class 说明正在定义一个新类。

(6) 关于 Public static void Main 的说明如下。

① Public 关键字是一个访问说明符。访问说明符确定程序其他部分如何访问类的成员。Public 表示类外面代码能访问该成员。因为程序开始执行 Main ( )将由它所在类的外面代码(操作系统)来调用,所以声明为 Public。

② static 表示允许在创建 Main( )所在类的对象之前调用 Main( )。由于 Main( )是在程序启动时调用的,所以这是必需的。

③ void 表明 Main( )不返回值。

(7) 代码块:{两个或多个语句}整体表示逻辑块。

(8) 处理语言错误。

由于语言写什么,C#编译器都试图了解源代码,因此所报告的信息往往不能反映问题的所在原因。当程序包含错误语法时,不必完全接受编译器表面的信息;它们可能是误导。为了找到问题的所在,需要"推测"错误信息。另外,需要查看错误信息所在代码前面的几行代码。有时,在实际问题所在位置之后几行才有错误报告。

3. 关键字和标识符

1) 关键字

C#语言中,关键字有 77 个,如表 3-1 所示。这些关键字不能作为变量、类或方法的名称来使用。

表 3-1　C#关键字

abstract	as	base	bool	break	byte	case	catch
char	checked	class	const	continue	decimal	default	delegate
do	double	else	enum	event	explicit	extern	false
finally	fixed	float	for	foreach	goto	if	implicit
in	int	interface	internal	is	lock	long	namespace
new	null	object	operator	out	override	params	private
protected	public	readonly	ref	return	sbyte	sealed	short
sizeof	stackalloc	static	string	struct	switch	this	throw
true	try	typeof	unit	ulong	unchecked	unsafe	ushort
using	virtual	volatile	void	while			

2) 标识符

在 C#中,标识符是分配给方法、变量或任何其他用户自定义对象的名称。

(1) 标识符由字母、数字和下划线组成,以字母、下划线打头。

(2) 标识符中大写和小写是有区别的。

(3) C#允许在关键字前面加上一个@作为有效标识符,但并不建议使用。

(4) 命名时尽量遵循以下规则。每个名字由几个单词连接而成;对于类名和方法,每

个单词开头大写；对于变量名，第一个字母不用大写；常量全部用大写标识符定义。

3.2.2 基本数据类型

C#有两种基本的类型：一种是值(value types)，一种是引用(reference types)。值包括简单类型(char、int 和 float)、枚举(enum)和结构(struct)。引用包括类(class)、界面(Interface)、代表(delegate)和数组(array)。

值与引用的不同之处在于：值直接存储它的数据内容，而引用存储对象的引用。值和引用的区别可以引出一个重要特性。值的变量和变量存储的数据是一一对应的，是唯一的。而引用则不然，引用中不同的变量可以引用同一个对象的实例。当其中一个变量改变实例的值时，其他引用这个实例的变量也会受到影响(当然，变量本身并没有改变，即地址没变)。如程序 3-2 所示。

程序 3-2 值和引用的区别

```
using System;
class Cvalue
{
  public int Value=0;
}
class Test
{
  Static void Main( )
  {
   int val1=0;
   int val2=val1;
   val2=123;
   CValue ref1=new VCalue( );
   CValue ref2=ref1;
   ref2.Value=123;
   Console.Writeline("Values:{0},{1}",val1,val2);
   Console.WriteLine("Refs: {0},{1}",ref1.Value,ref2.Value);
  }
}
```

输出结果如下：

```
Values: 0,123
Refs: 123,123
```

变量 val1 和变量 val2 互不影响，它们各自有自己的存储空间。而 ref2 复制了 ref1，所以它们引用了同一个对象的实例。当改变它们其中一个的时候，就会影响到另一个的值。

1. 值类型

表 3-2 给出了 C# 可以使用的值类型。

表 3-2　C# 中的值类型

类　型	字 节 数	描　　述	类　型	字 节 数	描　　述
byte	1	无符号字节	ulong	8	无符号大整数
sbyte	1	有符号的字节	float	4	浮点数
short	2	有符号的短字节	double	8	双精度数
ushort	2	无符号的短字节	decimal	8	固定精度数
int	4	有符号的整型数	string		Unicode 字符串
uint	4	无符号的整型数	char		Unicode 字符
long	8	有符号大整数	bool		Boolean 值

VB 和 C 程序员都可能会对 int 和 long 数据类型所代表的新范围感到惊讶。和其他的编程语言相比，在 C# 中，int 不再取决于一个机器中字(word)的大小，而 long 被设成 64 位。小数型是一种高精度、128 位数据类型，它用于金融和货币的计算。它所表示的范围为 1E−28～7.9E+28，具有 28～29 位有效数字。要注意，精度是以位数(digits)而不是以小数位(decimal places)表示的。运算准确到 28 个小数位的最大值。它的取信范围比 double 的还窄，但它更精确。因此，没有 decimal 和 double 之间的隐式转换，往一个方向转换可能会溢出，往另外一个方向转换可能会丢失精度。当定义一个变量并赋值给它时，使用 m 后缀以表明它是一个小数型，例如：

```
decimal decMyValue=1.0m;
```

如果省略了 m，在变量被赋值之前，它将被编译器认作 double 型。

2. 引用类型

和值类型相比，引用类型不存储它们所代表的实际数据，但它们存储实际数据的引用在 C# 中提供以下引用类型可以使用：对象类型、类类型、接口、代表元、字符串类型和数组。

1) 对象类型

对象类型是所有类型之母，它是其他类型最根本的基类。因为它是所有对象的基类，所以可把任何类型的值赋给它。

例如，一个整型：

```
object theObj=123;
```

2) 类类型

一个类类型可以包含数据成员、函数成员和嵌套类型。数据成员是常量、字段和事件。函数成员包括方法、属性、索引、操作符、构造函数和析构函数。类和结构的功能是非常相似的，但正如前面所述，结构是值类型，而类是引用类型。和 C ++相比，仅允许单继

承(不能拥有派生一个新对象的多重基类)。但是,C#中的一个类可以派生自多重接口。

3) 接口

一个接口声明一个只有抽象成员的引用类型。同C++中相似的概念为:一个结构的成员,且方法等于0。C#中一个接口实际所做的,仅仅存在着方法标志,根本就没有执行代码。这就暗示了不能实例化一个接口,只能实例化一个派生自该接口的对象。可以在一个接口中定义方法、属性和索引。当定义一个类时,可以派生自多重接口,但只能从仅有的一个类派生。

4) 代表元

一个代表元封装了一个具有一些标志的方法。代表元是类型安全和函数指针的安全版本(回调功能),可以同时在一个代表元实例中封装静态和实例方法。

5) 字符串类型

C#有一个用于操作字符串的基本字符串类型,字符串类直接派生自对象,且是被封装的,这意味着不能再从它派生类。字符串是预定义类 System String 的一个别名。

它的用法十分简单,例如:

```
string myString="some text";
```

合并字符串同样简单,例如:

```
string myString="some text"+"and a bit more";
```

如果想访问单个字符,则要访问下标,即:

```
char chFirst=myString[0];
```

当比较两个字符串是否相等时,简单地使用"=="比较操作符。

```
if(myString==yourString)
```

字符串是一个应用类型,比较时是比较值,而不是比较引用(内存地址)。

6) 数组

一个数组包含有通过计算下标访问的变量。所有包含于数组中且被当作元素的变量必须是同一类型。这种类型自然被称为"数组类型"。数组可以存储整数对象、字符串对象或者用户提出的任何对象。

用一个数组初始化器(array initializer)初始化的数组:

```
string [] arrLanguages={"C", "C++", "C# "};
```

该语句简写效果如下:

```
arrLanguages[0]="C"; arrLanguages[1]="C++"; arrLanguages[2]="C# ";
```

如果不想事先初始化一个数组,但知道了它的大小,可以以下面这种方式声明:

```
Int[,]myArr=new int[5,3];
```

如果数组的大小必须动态计算,用于数组创建的语句可以这样写:

```
int  nVar=5;
int  arrToo=new int[nVar];
```

3. 加框和消框

加框和消框是 c#类型系统的核心概念。通过允许一个值类型转换成类型对象或从类型对象转换成值类型,这种机制形成了值类型和引用类型之间的捆绑连接。

1) 加框转化

给一个值加框,指隐式地把任何值类型转换成类型对象。当一个值类型被加框时,一个对象实例就被分配,且值类型的值被复制给新的对象。

看以下例子:

```
int nFunny=2000;
object oFunny=nFunny;
```

第二行的赋值暗示调用一个加框操作。nFunny 整型变量被复制给 oFunny 对象。现在整型变量和对象变量都同时存在于栈中,但对象的值留在堆中。

2) 消框转换

和加框相比,消框是显示操作,必须告诉编译器要从对象中抽取哪一种数据类型。当执行消框操作时,C#检测所请求的值类型实际上存储在对象事例中。经过成功的校验,该值被消框。

继续刚才的例子,进行消框:

```
int nFunny=2000;
object oFunny=nfunny;
int nNotSoFunny=(int)oFunny;
```

3.2.3 运算符

C#提供大量的运算符,使程序设计员对表达式的结构和计算进行具体控制。C#有 5 种基本运算符:算术运算符、关系运算符、逻辑运算符、赋值运算符和位运算符。

1. 5 种基本运算符

1) 算术运算符

C#定义了+(加)、-(减)、*(乘)、/(除)、%(求余)、++(递增)和--(递减)7 种算术运算符。

关于算术运算符的几点说明如下。

(1) +、-、*、/ 可以应用于任何内部数字数据类型。

(2) %可以应用于整型数据类型,也可以应用于浮点数据类型。

(3) 关于++、--位置所引起的区别:当递增运算符(或递减运算符)在操作数之前时,C#在获得操作数的值用于表达式其余部分的运算之前,先进行递增(或递减)运算。如果运算符在操作数的后面,那么 C#先取得该操作数用于其他计算,然后再递增(或递减)它。

2）关系运算符

C＃定义了＝＝（等于）、!＝（不等于）、＞（大于）、＞＝（大于等于）、＜（小于）和＜＝（小于等于）6 种关系运算符。

（1）＝＝和!＝可以比较任意对象是否相等或不相等。

（2）＞、＞＝、＜和＜＝只能用于那些支持次序关系的类型，因此支持任意数字类型，但不支持 bool 类型值。

（3）经关系运算后的运算结果为 bool 类型的值。

3）逻辑运算符

C＃定义了 &（与）、|（或）、!（非）、^（异或）、&&（短路与）和||（短路或）6 种逻辑运算符。

（1）对于逻辑运算符，其操作数是 bool 类型，并且逻辑运算结果也为 bool 类型。

（2）&& 的第一个操作数为 0，不计算第二个操作数，结果为假。||的第一个操作数为 1，不计算第二个操作数，结果为真。

（3）& 和|不管什么情况下都要计算两个操作数，它们是按位运算，都是把参加运算的数以二进制形式进行运算。

4）赋值运算符

（1）“＝”运算符。

简单赋值：var1＝expression;

赋值链：var1＝var2＝var3＝expression;

（2）复合赋值。

对于所有二元运算符，都有复合赋值运算符。通常简写形式为：

```
var op=expression;
```

复合运算符比普通运算符更简单，由于操作数只计算一次，因此能产生更高效的可执行代码。

5）位运算符

C＃定义了 &（按位与）、|（按位或）、^（按位异或）、～（按位非）、＜＜（左移）和＞＞（右移）6 种位运算符。

（1）&、|、^和～运算符说明如下。

&0：用于清 0 操作。

|1：用于置 1 操作。

^：0^varb＝varb;，1^varb＝～varb;。

～：各位求反，用于求补码和反码运算。

（2）移位运算符＜＜、＞＞。

左移：右边补 0。

右移：无符号直接在左边补 0；有符号正数左边补 0，有符号负数左边补 1。

（3）对于左移和右移，被移走的位被丢弃，因此移位运算不能循环，没有办法获得被移出的位。

(4) 移位运算符可以作为整数乘 2 或除 2 的快速计算方法。＜＜(加倍)和＞＞(减半)只能在不会将两端位移出时使用。

2. 运算符的优先级

C#运算符的优先级从高到低如下。

(1) ( )、[ ]、＋＋(后置)、--(后置)、checked、new、sizeof、typeof、unchecked

(2) !、~、(cast)、＋(一元)、－(一元)、＋＋(前置)、－－(前置)

(3) *、/、%

(4) ＋、－

(5) ＜＜、＞＞

(6) ＞、＞＝、＜、＜＝、is

(7) ＝＝、!＝

(8) &

(9) ^

(10) |

(11) &&

(12) ||

(13) ?:

(14) ＝、op＝

3.2.4　程序控制语句

编写程序时最重要的就是要了解应用程序的控制结构，控制结构是通过控制语句来实现的，控制语句包括条件语句、循环语句和跳转语句。

1. 条件语句

C#支持三种基本的条件语句：if 语句、? 条件和 switch 语句。

1) if 语句

if 有如下三种语句。

(1)

```
if(条件)
{
  语句;
}
```

(2)

```
if(条件)
{
    语句;
}
else
```

```
{
  语句 1;
}
```

(3)

```
if(条件 1)
{
   语句 1;
}
else if(条件 2)
{
  语句 2
}
⋮
else
{
  语句 N;
}
```

2) ? 条件

```
条件?语句 1: 语句 2
等价于
  if(条件)
  {
    语句;
  }
  else
  {
    语句 1;
  }
```

3) switch 语句

```
switch(条件)
{
case 常量表达式 1: 语句 1;break;
case 常量表达式 2: 语句 2;break;
case 常量表达式 3: 语句 3;break;
⋮
default:语句 N;break;
}
```

控制表达式所允许的数据类型为 sbyte、byte、short、ushort、uint、long、ulong、char、string 或者枚举类型。只要使其他不同数据类型能隐式转换成上述的任何类型,也可以将它作为控制表达式。

switch 语句按以下顺序执行。

(1) 控制表达式求值。

(2) 如果 case 标签后的常量表达式符合控制语句所求出的值,内含语句被执行。

(3) 如果没有常量表达式符合控制语句,在 default 标签内的内含语句被执行。

(4) 如果没有一个符合 case 标签,且没有 default 标签,控制转向 switch 语段的结束端。

2. 循环语句

当重复执行某些语句或语段时,依据当前不同的任务,C#提供了 4 个不同的循环语句:for、foreach、while 和 do。

1) for 语句

当预先知道一个内含语句要执行多少次时,可以使用 for 语句。当条件为真时,常规语法允许重复地执行内含语句(和循环表达式)。

for 语句格式如下。

```
for (初始化;条件;循环)
{
  语句;
}
```

注意:*初始化、条件和循环都是可选的。如果忽略了条件,就产生一个死循环,要用跳转语句(break 或 goto)才能退出。*

例如:

```
for (;;)
{
 break;        //由于某些原因
}
```

另外一个重点是,可以同时加入多条由逗号隔开的语句到 for 循环的三个参数中。例如,可以初始化两个变量,拥有 3 个条件语句,并重复 4 个变量。

2) foreach 语句

已经在 Visual Basic 语言中存在了很久的一个功能是,通过使用 foreach 语句收集枚举。C#通过 foreach 语句,也有一个用来收集枚举的命令。

```
foreach (表达式中的类型标识符)
{
    语句;
}
```

循环变量由类型和标识符声明,且表达式与收集相对应。循环变量代表循环正在为之运行的收集元素。不能赋一个新值给循环变量,也不能把它当作 ref 或 out 参数。

类必须支持具有 GetEnumerator()名字的方法,而且由它返回的结构、类或者接口必须具有 public 方法 MoveNext()和 public 属性 Current。

例如，读所有环境变量，如程序 3-3 所示。

程序 3-3　利用 foreach 读取环境变量

```
using System;
using System.Collections;
class EnvironmentDumpApp
{
  public static void Main()
  {
    IDictionary envvars=Environment.GetEnvironmentVariables();
     Console.WriteLine("There are {0} environment variables declared",envvars.Keys.
Count);
    foreach(String strKey in envvars.Keys)
    {
     Console.WriteLine("{0}={1}",strKey,envvars[strKey].ToString());
    }
  }
}
```

对 GetEnvironmentVariables 的调用返回一个 IDictionary 类型接口，它是由.NET 框架中的许多类实现的字典接口。通过 IDictionary 接口，可以访问两个收集：Keys 和 Values。在这个例子里，在 foreach 语句中使用 Keys，接着查找基于当前键值。

当使用 foreach 时，要注意一个问题：当确定循环变量的类型时，应该格外小心。选择错误的类型且没有受到编译器的检测，它会在运行时受检测，将引发一个异常。

3）while 语句

```
While(条件)
{
  语句;
}
```

条件语句也是一个布尔表达式，控制语句被执行的次数。可以使用 break 和 continue 语句来控制 while 语句中的执行语句，它的运行方式同在 for 语句中的完全相同。

4）do 语句

C＃最后可利用的循环语句是 do 语句。它与 while 语句十分相似，仅当经过最初的循环之后，条件才被验证。

```
do
{
 语句;
}
while(条件);
```

do 语句保证语句至少被执行过一次，而且只要条件求值等于真，它们将继续被执行。

通过使用 break 语句，可以迫使运行退出 do 语句。如果想跳过这一次循环，可以使用 continue 语句。

3. 跳转语句

跳转语句包括 break、continue、return 和 goto 等语句。

1）break 语句

（1）循环中遇到 break 语句时，循环终止，而程序控制流继续运行循环后面的语句。

（2）内层循环中 break 语句只终止该循环过程，而外层循环未受影响。

（3）break 语句可以与任意 C＃循环一起使用，其中包含有意的无限循环。

（4）一个循环中可以有多余的 break 语句，但此时必须谨慎，太多的 break 语句可能拆散代码结构。

（5）终止 switch 语句的 break 语句只影响该 switch 语句而不影响包含它的外层循环。

2）continue 语句

（1）continue 语句是跳出本次循环中尚未执行的语句进入下一次循环。

（2）while 和 do-while 语句中，continue 语句将控制流直接跳到条件表达式，然后继续循环过程。

（3）for 循环中，先计算循环的迭代表达式，执行条件表达式，然后继续循环过程。

3）return 语句

reture 语句将控制权返回给方法的调用者，它也可以用来返回值。

4）goto 语句

没有哪种程序设计环境会要求使用 goto 语句，该语句不是保证语言的完整性所必要的。

（1）goto 语句要求有操作标签。标签是一个后面带冒号"："的合法的 C＃标识符。另外，标签必须与使用它的 goto 语句处于相同方法中。

（2）goto 语句的一种较好的使用是用于从深层嵌套中跳出。

（3）goto 语句也可以用来跳转 switch 中的 case 语句或 default 语句。switch 中的 case 语句和 default 语句其实就是标签，因此它们可以作为 goto 的目标位置。goto 语句必须在 switch 内部执行，即不能使用 goto 语句跳转到 switch 语句中。

3.2.5　类

1. 构造函数和析构函数

在访问一个类的方法、属性或任何其他内容之前，第一条执行的语句是包含有相应类的构造函数。甚至不用写一个构造函数，也会有一个默认的构造函数提供给用户。

```
class TestClass
{
 Public TestClass: base(){ }              //有编译器提供
}
```

一个构造函数总是和它的类名相同，但是，它没有声明返回类型。总之，构造函数总是 public 的，可以用它们来初始化变量。

```
public TestClass()
{
//初始化变量代码等
}
```

如果类仅包含静态成员（能以类型调用，而不是以实例调用的成员），可以创建一个 private 的构造函数。

```
private TestClass()
{
  ⋮
}
```

private 意味着从类的外面不能访问该构造函数。所以，它不能被调用，且没有对象可以自该类定义被实例化。

一个 C/C++程序员可能习惯于给初始化写一个附加的方法，因为在构造函数中没有返回值。当然，尽管在 C# 中也没有返回值，但可以引发一个自制的异常，以从构造函数中获得返回值。

当要释放引用的资源时，写一个可以被显式地调用来释放这些资源的方法。若在析构函数（以类名的前面加"～"的方式命名）中释放引用的资源，需要写一个附加的方法。

```
public ~TestClass()
{
//清除
}
```

写一个附加方法的原因是垃圾收集器，它在变量超出范围后并不会立即被调用，而仅当间歇期间或内存条件满足时才被触发。当锁住资源的时间大于所计划的时间时，它就会发生。因此，一个显式的释放方式，最好同样能从析构函数中调用。

```
public void Release()
{
//释放所有宝贵的资源
}
Public ~TestClass
{
Release();
}
```

调用析构函数中的释放方法并不是必要的，因为垃圾收集器会留意释放对象。但不忘记释放引用的资源是一种良好的习惯。

2. 方法

既然对象能正确地初始化和结束，所剩下来的工作就是往类中增加功能。在大多数情况下，功能的主要部分在方法中得到实现。

1) 方法参数

因为方法要处理更改数值，将值传递给方法，并从方法获得返回值，所以将涉及由传递值和为调用者获取返回结果所引起的三方面的问题：输入参数、引用参数和输出参数。

(1) 输入参数。

一个输入参数通过值传递一个变量给一个方法，方法的变量被调用者传递来的值的一个备份初始化，如程序 3-4 所示。

程序 3-4　通过值传递参数

```
using System;
public class SquareSample
{
  Public int CalcSquare(int nSideLength)
  {
    return nSideLength *  nSideLength;
  }
}
class SquareApp
{
  public static void Main()
  {
    SquareSample sq=new SquareSample();
    Console.WriteLine(sq.CalcSquare(25).ToString());
  }
}
```

因为是传递值而不是引用一个变量，所以当调用方法时，可以使用一个常量表达式。整型结果被传回给调用者作为返回值，它没有保存到中间变量中就被立即显示到屏幕上。

输入参数用对于 C/C ++程序员早已习惯的工作方式工作。如果用户是一个 VB 程序员，应注意不能被编译器处理的隐式 ByVal 或 ByRef，如果没有设定，参数总是用值传递。

这点似乎与前面所陈述的有所冲突：对于一些变量类型，用值传递实际上意味着用引用传递。COM 中的东西就是接口，每一个类可以拥有一个或多个接口。一个接口只不过是一组函数指针，它不包含数据，重复该数组会浪费很多内存资源。所以，仅有开始地址被复制给方法，它作为调用者，仍然指向接口的相同指针。这就是对象用值传递一个引用的原因。

(2) 引用参数。

尽管可以利用输入参数和返回值建立很多方法,但要传递值并原地修改它(也就是在相同的内存位置),这时用引用参数比较方便。

```
void myMethod(ref int nInOut)
```

因为传递了一个变量给方法,变量必须被初始化,否则编译器会报警。

程序 3-5 显示了如何用一个引用参数建立一个方法,通过引用传递参数。

程序 3-5　通过引用传递参数

```
using System;
public class SquareSample
{
  public void CalcSquare(ref int nOne4All)
  {
      nOne4All * =nOne4All;
  }
}
class SquareApp
{
  public static void Main()
    {
      SquareSample sq=new SquareSample();
      int nSquaredRef=20//一定要初始化
      sq.CalcSquare(ref nSquaredRef);
      Console.WriteLine(nSquaredRef .ToString());
    }
}
```

正如所看到的,所有要做的就是给定义和调用都加上 ref 限定符。因为变量通过引用传递,可以用它来计算出结果并传回该结果。但是,在现实的应用程序中,强烈建议要用两个变量,一个作为输入参数,另一个作为引用参数。

(3) 输出参数。

传递参数的第三种选择就是把它设为一个输出参数。一个输出参数仅用于从方法传递回一个结果。它和引用参数的另一个区别在于:调用者不必先初始化变量才调用方法。如程序 3-6 所示。

程序 3-6　输出参数的应用

```
using System;
public class SquareSample
{
public void CalcSquare(int nSideLength,out int nSquared)
  {
```

```
    nSquared=nSideLength * nSideLength;
  }
}
class SquareApp
{
  public static void Main()
  {
    SquareSample sq=new SquareSample();
    int nSquared;                     //不必初始化
    sq.CalcSqare(15,outnSquared);
    Console.WriteLine(nSquared.ToString());
  }
}
```

2）改写方法

面向对象设计的重要原则是多态性。多态性意味着：当基类程序员已设计好用于改写的方法时，在派生类中，可以重定义（改写）基类的方法。基类程序员可以用 virtual 关键字设计方法。

```
Virtual void CanBOverridden()
```

当从基类派生时，要做的就是在新方法中加入 override 关键字。

```
Override void CanBOverridden()
```

当改写一个基类的方法时，不能改变方法的访问属性。

除了改写基类方法外，还有另一个更重要的改写特性。当把派生类强制转换成基类类型并接着调用虚拟方法时，被调用的是派生类的方法而不是基类的方法。

```
((BaseClass)DerivedClassInstance).CanBoverridden();
```

为了演示虚拟方法的概念，程序 3-7 显示了如何创建一个三角形基类，它拥有一个可以被改写的成员方法（ComputeArea）。

程序 3-7　创建三角形基类

```
using System;
class Triangle
{
  Public virtual double ComputeArea(int a,int b,int c)
  {
    double s=(a+b+c)/2.0;
    double dArea=Math.Sqrt(s*(s-a)*(s-b)*(s-c));
    return dArea;
  }
}
```

```
class RightAngledTriangle:Triangle
{
  public override double ComputeArea(int a,int b,int c)
  {
    double dArea=a * b/2.0;
    return dArea;
  }
}
class TriangleTestApp
{
  Public static void Main()
  {
    Triangle tri=new Triangle();
    Console.WriteLine(tri.ComputeArea(2,5,6));
    RightAngledTriangle rat=new RightAngledTriangle();
    Console.WriteLine(rat.ComputeArea(3,4,5));
  }
}
```

基类 Triangle 定义了方法 ComputeArea。它采用三个参数，返回一个 double 结果，且具有公共访问性。从 Triangle 类派生出的是 RightAngledTriangle，它改写了 ComputeArea 方法，并实现了自己的面积计算公式。两个类都被实例化，且在命名为 TriangleTestApp 的应用类的 Main 方法中得到验证。

其中 class RightAngledTriangle：Triangle 在类语句中，冒号（：）表示 RightAngledTriangle 从类 Triangle 派生。这是必须要做的，以让 C＃知道用户要把：Triangle 当作 RightAngledTriangle 的基类。当仔细观察直角三角形的 ComputeArea 方法时，会发现第三个参数并没有用于计算。但是，利用该参数可以验证是否是“直角”，如程序 3-8 所示。

程序 3-8　显示了如何调用基类

```
class RightAngledTriangle:Triangle
{
  public override double ComputeArea(int a,int b,int c)
  {
      const double dEpsilon=0.0001;
      double dArea=0;
      if(Math.Abs((a * a+b * b-c * c))>dEpsilon)
      {
          dArea=base.ComputeArea(a,b,c);
      }
      else
```

```
        {
          dArea=a * b/2.0;
        }
      return dArea;
    }
  }
```

该检测简单地利用了毕达哥拉斯公式，对于直角三角形，检测结果必须为 0。如果结果不为 0，类将调用基类的 ComputeArea 来实现。

```
dArea=base.ComputeArea(a,b,c);
```

上例说明了通过显式地利用基类的资格检查，可以轻而易举地调用基类实现改写方法。当需要实现在基类中的功能而不愿意在改写方法中重复它时，这是非常有帮助的。

3）方法屏蔽

重定义方法的一个不同之处就是要屏蔽基类的方法。当从别人提供的类派生类时，这个功能特别有用。在下例中假设 BaseClass 是由他人所写，而用户需要从它派生出 DerivedClass。如程序 3-9 所示。

程序 3-9　利用 DerivedClass 实现了一个没有包含于 BaseClass 中的方法

```
using System;
class BaseClass
{
}
class DerivedClass:BaseClass
{
 public void TestMethod()
 {
  Console.WriteLine("DerivedClass::TestMethod");
 }
}
class TestApp
{
 public static void Main()
 {
  DerivedClass test=new DerivedClass();
  Test.TestMethod();
 }
}
```

若有修饰符 new，则可以告诉编译器不必重写派生类或改变使用到派生类的代码，用户的方法将能屏蔽新加入的基类方法。如程序 3-10 所示。

程序 3-10 如何运用 new 修饰符及如何屏蔽基类方法

```
class BaseClass
{
public void TestMethod()
  {
   Console.WriteLine("BaseClass::TestMethod");
  }
}
class DerivedClass:BaseClass
{
new public void TestMethod()
{
  Console.WriteLine("DerivedClass::TestMethod");
}
}
```

使用了附加的 new 修饰符，编译器就知道重定义了基类的方法，它应该屏蔽原来的基类方法。但是，如果按以下方式编写：

```
DerivedClass test=new DerivedClass();
((BaseClass)test).TestMethod();
```

基类方法的实现就被调用了。这种行为不同于改写方法，后者保证了大部分派生方法能获得调用。

3. 类属性

有两种途径揭示类的命名属性，通过域成员或者通过属性。前者是作为具有公共访问性的成员变量而被实现的，后者并不直接回应存储位置，只是通过存取标志(accessors)被访问。

当想读出或写入属性的值时，存取标志限定了被实现的语句。用于读出属性值的存取标志记为关键字 get，而要修改属性值的读写符标志记为 get，如程序 3-11 所示。

程序 3-11 实现属性存取标志

```
using System;
public class House
{
 private int m_nSqFeet;
 public int SquareFeet
 {
  get
  {
   return m_nSqFeet;
  }
```

```
      set
      {
       m_nSqFeet=value;
      }
    }
  }
  class TestApp
  {
  public static void Main()
  {
   House myHouse=new House();
   myHouse.SquareFeet=250;
   Consol.WriteLine(myHouse.SquareFeet);
  }
  }
```

House 类有一个命名为 SquareFeet 的属性,它可以被读写。实际的值存储在一个可以从类内部访问的变量中,如果想当作一个域成员重写,所要做的就是忽略存取标志而把变量重新定义为:

```
public int SquareFeet;
```

对于一个简单的变量可以这样定义。但是,如果想要隐藏类内部存储结构的细节时,就应该采用存取标志。在这种情况下,set 存取标志给值参数中的属性传递新值。

除了能够隐藏实现细节外,还可自由地限定各种操作。get 和 set 允许对属性进行读写访问;get only 只允许读属性的值;set only 只允许写属性的值。

4. 索引

使用 C♯的索引功能可以像访问数组那样使用索引访问类。

语法基本如下:

```
属性修饰符声明{声明内容}
```

具体的例子为:

```
public string this[int nIndex]
{
 get{…}
 set{…}
}
```

索引返回或按给出的 index 设置字符串。它没有属性,但使用了 Public 修饰符。声明部分由类型 string 和 this 组成用于表示类的索引。

get 和 set 的执行规则和属性的规则相同,只存在一个差别,那就是,几乎可以任意定义大括号中的参数。限制为必须至少规定一个参数,允许 ref 和 out 修饰符。

this 关键字确保一个解释。索引没有用户定义的名字,this 表示默认接口的索引。

如果类实现了多个接口,可以增加更多个由 InterfaceName、this 说明的索引。

5. 事件

当写一个类时,有时有必要让类的客户知道一些已经发生的事件。如果是一个具有多年编程经验的程序员,可以有很多解决办法,包括用回调的函数指针和用 ActiveX 控件的事件接收(event sinks)。下面将介绍另外一种把客户代码关联到类通知的方法——使用事件。

事件既可以被声明为类域成员(成员变量),也可以被声明为属性。两者的共性为:事件的类型必定是代表元,而函数指针原型和 C# 的代表元具有相同的含义。

每一个事件都可以被一个或多个客户占用,且客户可以随时关联或取消事件。可以以静态或者以实例方法定义代表元,而后者很受 C ++程序员的欢迎。程序 3-12 显示如何在类中实现事件处理。

程序 3-12 在类中实现事件处理

```
using System
//向前声明
public delegate void EventHandler(string strText);
class EventSource
{
  public event EventHandler TextOut;
  public void TriggerEvent()
  {
    if(null!=TextOut)
    TextOut("Event triggered");
  }
}
class TestApp
{
  public static void Main()
  {
    EventSource evsrc=new EventSource();
    event.TextOut+=new EventHandler(CatchEvent);
    evsrc.TriggerEvent();
    event.TextOut-=new EventHandler(CatchEvent);
    evsrc.TriggerEvent();
    TestApp theApp=new TestApp();
    Evsrc.TextOut+=new EventHandler(theApp.InstanceCatch);
    Evsrc.TriggerEvent();
  }
public static void CatchEvent(string strText)
{
  Console.WriteLine(strText);
}
```

```
public void InstanceCatch(string strText)
{
 Console.WriteLine("Instance"+ strText);
}
}
```

TestApp 类包含了 Main 方法，也包含了另外两个方法，它们都具备事件所必需的信号。其中一个方法是静态的，而另一个是实例方法。

EventSource 被实例化，而静态方法 CatchEvent 被关联上了 TextOut 事件。

```
Evsrc.TextOut+=new EventHandler(theApp.InstanceCatch);
```

从现在起，当事件被触发时，该方法被调用。如果对事件不再感兴趣，可以取消关联。

```
event.TextOut-=new EventHandler(CatchEvent);
```

为了证明事件处理函数也和实例方法一起工作，余下的代码建立了 TestApp 的实例，并关联事件处理方法。

在 ASP. NET 中会经常涉及事件和代表元。

3.3　ASP.NET 控件

ASP. NET 的强大功能和便捷离不开工具箱控件的支持。控件可分为标准控件、数据控件、验证控件、导航控件、登录控件、WebParts 控件、HTML 控件和水晶报表控件等。用户可以用两种方法添加控件：从工具箱直接拖放控件到 Web 页面；通过添加相应代码添加控件。

3.3.1　Web 窗体的标准控件

1. 单选控件和单选组控件(RedioButton 和 RedioButtonList)

单选控件的选择只能有一种。凡是从有限种可能性中，只能选择一种结果时，都可以用单选控件来实现。单选控件通过 Checked 属性来判断是否被选中。多个单选控件之间可以存在着联系，此时单选控件通过 GroupName 属性来对这些控件约束和联系，它用来指明多个单选控件是否是在同一条件下的选择项。对 GroupName 相同的多个单选控件之间，只能有一个被选中。单选控件最常用的事件是 CheckChanged。当控件选中状态改变时，将会激发该事件。下面来看两个应用实例。

1) 单选控件应用实例

在工具箱中选中单选控件，拖曳三个单选按钮到窗体上，然后利用属性窗口，设置单选按钮如表 3-3 所示的属性。

表 3-3 单选按钮控件属性

属性	值	属性	值
Check	False	GroupName	color
ID	rbChoose1 rbChoose2 rbChoose3	Text	红色 蓝色 绿色

对每个单选按钮编写相应的代码，如程序 3-13 所示。

程序 3-13 单选按钮对应的代码

```
protected void rbChoose1_CheckedChanged(object sender, EventArgs e)
    {
        //如果选中第一个单选按钮，改变不同的背景色
        if (this.rbChoose1.Checked==true)
        {
            this.rbChoose1.ForeColor=Color.Red;
            this.rbChoose2.ForeColor=Color.Transparent;
            this.rbChoose3.ForeColor=Color.Transparent;
        }
    }
    protected void rbChoose2_CheckedChanged(object sender, EventArgs e)
    {
        //如果选中第二个单选按钮，改变不同的背景色
        if (this.rbChoose2.Checked==true)
        {
            this.rbChoose1.ForeColor=Color.Transparent;
            this.rbChoose2.ForeColor=Color.Blue;
            this.rbChoose3.ForeColor=Color.Transparent;
        }
    }
    protected void rbChoose3_CheckedChanged(object sender, EventArgs e)
    {
        //如果选中第三个单选按钮，改变不同的背景色
        if (this.rbChoose3.Checked==true)
        {
            this.rbChoose1.ForeColor=Color.Transparent;
            this.rbChoose2.ForeColor=Color.Transparent;
            this.rbChoose3.ForeColor=Color.Green;
        }
    }
```

运行本程序的效果如图 3-16 所示。

2）单选组控件应用实例

在工具箱中选中 RadioButtonList 控件，将其拖放到窗体上。然后利用属性窗口设置属性，如表 3-4 所示。



图 3-16　单选按钮效果图

表 3-4　单选组控件属性

属　　性	值	
AutoPostBack	True	
ID	rblBackColor	
Items	Text 红色背景 蓝色背景 绿色背景	Value 3 4 5

同时在窗口上放置一个标签控件，ID 属性设置为 lbText，Text 属性设置为“实验文本”。

窗体的 HTML 代码如程序 3-14 所示。

程序 3-14　单选组控件对应的 HTML 代码

```
<form id="form1" runat="server">
  <div>
   <asp:RadioButtonList ID="rblBackColor" runat="server" AutoPostBack="True" Style="z
- index: 100; left: 167px; position: absolute; top: 84px" OnSelectedIndexChanged
="rblBackColor_SelectedIndexChanged">
           <asp:ListItem Value="3">红色背景</asp:ListItem>
           <asp:ListItem Value="4">蓝色背景</asp:ListItem>
           <asp:ListItem Value="5">绿色背景</asp:ListItem>
   </asp:RadioButtonList>
   <asp:Label ID="lbText" runat="server" Style="z-index: 102; left: 337px; position:
absolute;top: 115px" Text="实验文本"></asp:Label>
   </div>
</form>
```

双击 RadioButtonList 控件，对 SelectedIndexChanged 事件进行编程，使得对 RadioButtonList 中每个单选按钮的每次选择，都同时改变标签文本的背景色。如程序 3-15 所示。

程序 3-15　单选组控件实例对应的.CS 代码

```
protected void rblBackColor_SelectedIndexChanged(object sender, EventArgs e)
{
```

```
        //如果选中了第一个按钮,则改变标签文本相应背景色为红色
        if (this.rblBackColor.SelectedIndex==0)
        {
            this.lbText.BackColor=Color.Red;
        }
        //如果选中了第二个按钮,则改变标签文本相应背景色为蓝色
        else if (this.rblBackColor.SelectedIndex==1)
        {
            this.lbText.BackColor=Color.Blue;
        }
        //如果选中了第三个按钮,则改变标签文本相应背景色为绿色
        else if (this.rblBackColor.SelectedIndex==2)
        {
            this.lbText.BackColor=Color.Green;
        }
}
```

程序执行效果如图 3-17 所示。

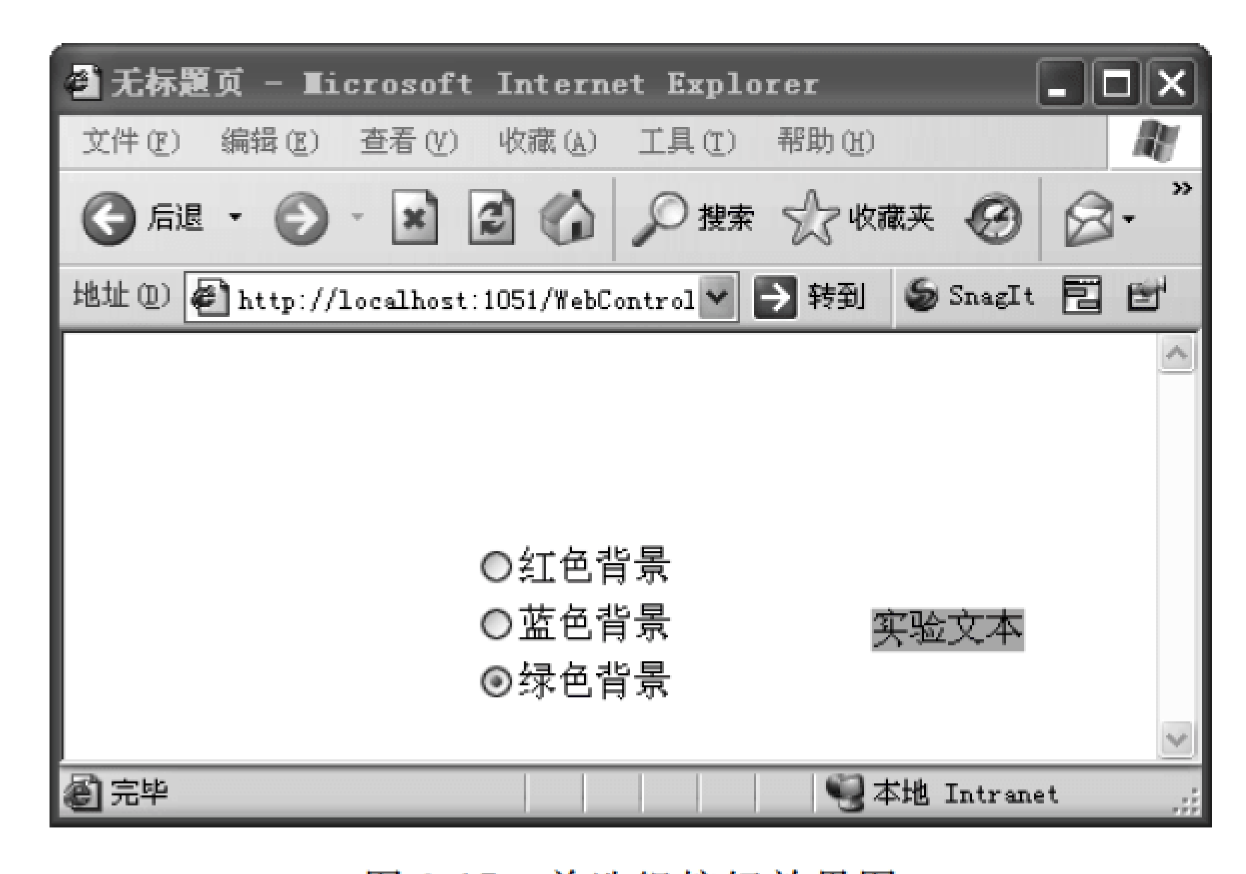


图 3-17　单选组按钮效果图

2. 复选控件和复选组控件(CheckBox 和 CheckBoxList)

当录入的信息具有很多种可能时,单选控件就不再满足需要了。此时,可以使用 CheckBox 复选框控件,使多种可能性可以同时被选择。复选框控件同样依靠 Checked 属性来判断是否被选中。下面是复选控件的应用实例。

在工具箱中选择复选控件,拖拉三个复选框至窗体上,同时放置一个 Label 控件,设置其 Text 属性为"示例文本",ID 为 lbText,利用属性窗口来设置其他控件属性,如表 3-5 所示。

在设计窗体页面,逐个双击复选按钮。每次双击都会添加对应复选框的 CheckChanged 事件。用户也可以在复选按钮属性页中添加 CheckChanged 事件。如程序 3-16 所示。

表 3-5　复选框控件属性

属　　性	值	属　　性	值
Checked	False	AutoPostBack	True
ID	cbChoose1 cbChoose2 cbChoose3	Text	文字具有上划线 文字具有下划线 文字具有删除线

程序 3-16　复选控件实例对应的.CS 代码

```
protected void cbChoose1_CheckedChanged(object sender, EventArgs e)
    {
        // 如果被选中,则标签文本显示上划线
        if (this.cbChoose1.Checked==true)
        {
            this.lbText.Font.Overline=true;
        }
        // 否则去掉标签文本的上划线
        else
        {
            this.lbText.Font.Overline=false;
        }
    }
    protected void cbChoose2_CheckedChanged(object sender, EventArgs e)
    {
        // 如果被选中,则标签文本显示下划线
        if (this.cbChoose2.Checked==true)
        {
            this.lbText.Font.Underline=true;
        }
        //否则去掉标签文本的下划线
        else
        {
            this.lbText.Font.Underline=false;
        }
    }
    protected void cbChoose3_CheckedChanged(object sender, EventArgs e)
    {
        // 如果被选中,则标签文本显示删除线
        if (this.cbChoose3.Checked==true)
        {
            this.lbText.Font.Strikeout=true;
        }
        // 否则去掉标签文本的删除线
        else
```

```
            {
                this.lbText.Font.Strikeout=false;
            }
    }
```

程序执行效果如图 3-18 所示。

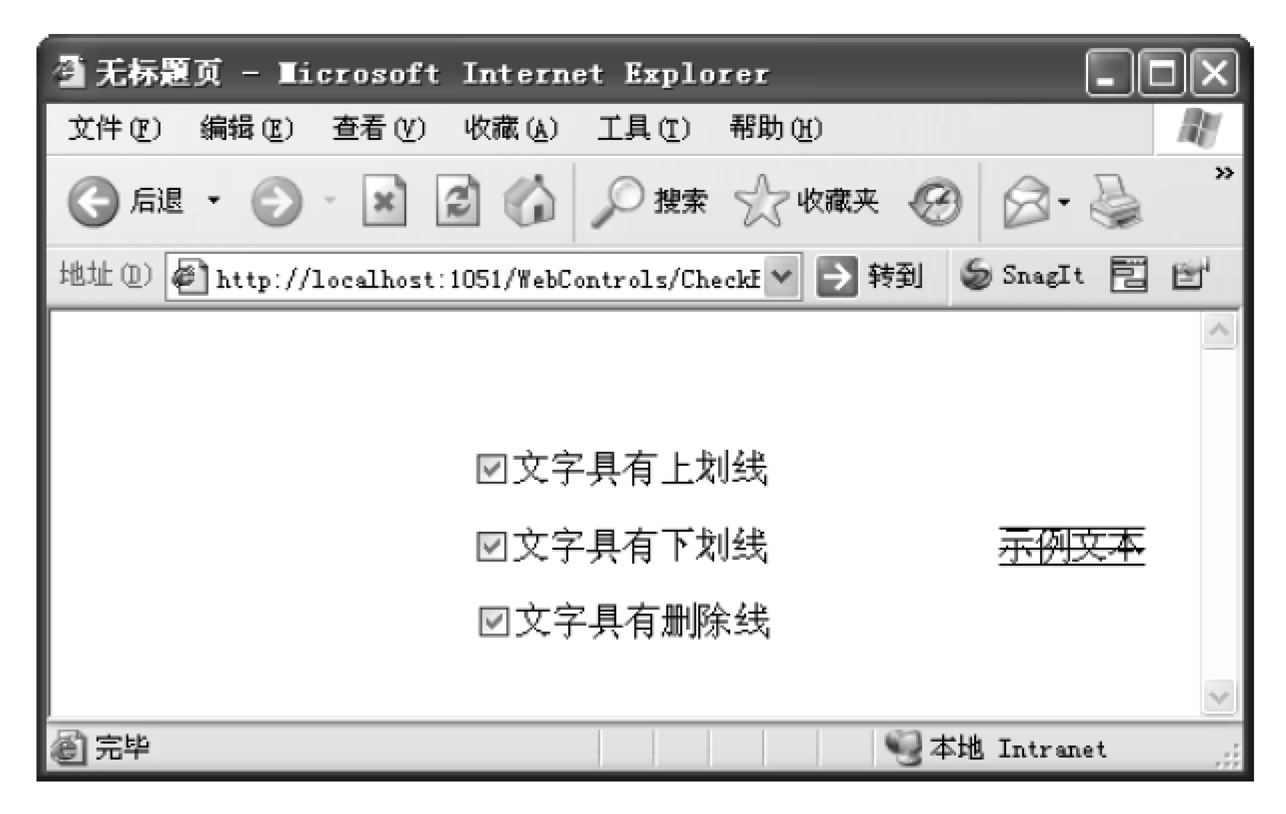


图 3-18 复选框控件效果图

3. 列表控件(DropDownList、ListBox 和 BulletedList)

列表控件可以在一个控件框内为用户提供多个选项。最常用的列表控件是 DropDownList,其次还有 ListBox、BulletedList 控件可选择。DropDownList、ListBox 和 BulletedList 控件都以 HTML 的＜select＞标签来显示到浏览器上,列表控件中的项由 ListItem 表示,Text 指定代表列表控件项的文本,Value 则是与该项关联的值。ListItem 还有一个 Selected 的属性,它表示该项是否被选中。下面是列表控件应用实例。

在工具箱中选择 DropDownList,将其拖放至窗体上,设置属性如表 3-6 所示。

在工具箱中选择 ListBox,将其拖放至窗体上,利用属性窗口设置属性如表 3-7 所示。

表 3-6 DropDownList 控件属性

属　　性	值	
ID	ddMusic	
AutoPostBack	True	
Item	Text	Value
	儿歌	0
	流行音乐	1
	古典音乐	2
	爵士乐	3
	蓝调音乐	4

表 3-7 ListBox 控件属性

属　　性	值	
ID	lbChoose	
AutoPostBack	True	
Item	Text	Value
	小说	0
	散文	1
	议论文	2
	诗歌	3
	译著	4
	杂文	5

在工具箱中选择 BulletedList，将其拖放至窗体上，利用属性窗口设置属性如表 3-8 所示。

表 3-8　BulletedList 控件属性

<table>
<tr><th>属　性</th><th colspan="2">值</th></tr>
<tr><td>ID</td><td colspan="2">blListChoose</td></tr>
<tr><td rowspan="8">Item</td><td>Text</td><td>Value</td></tr>
<tr><td>篮球</td><td>A</td></tr>
<tr><td>拳击</td><td>B</td></tr>
<tr><td>体操</td><td>C</td></tr>
<tr><td>划艇</td><td>D</td></tr>
<tr><td>滑雪</td><td>E</td></tr>
<tr><td>射击</td><td>F</td></tr>
<tr><td>田径</td><td>G</td></tr>
</table>

在设计窗体上，同时放置一个 Label 控件，ID 为 lbText。在示例程序中，将使用 Label 控件来显示用户选择了什么内容。如程序 3-17 所示。

程序 3-17　列表控件实例所对应的.CS 代码

```
 protected void ddlMuisc_SelectedIndexChanged(object sender, EventArgs e)
{
    this.lbText.Text="您在 DropDownList 控件中选择了："+this.ddlMuisc.SelectedItem.
Text;
    this.lbText.ForeColor=Color.Blue;
}
protected void lbChoose_SelectedIndexChanged(object sender, EventArgs e)
{
    this.lbText.Text="您在 ListBox 控件中选择了："+this.lbChoose.SelectedItem.Text;
    this.lbText.ForeColor=Color.Blue;
}
protected void blListChoose_Click(object sender, BulletedListEventArgs e)
{
    this.lbText.Text="您在 BulletedList 控件中选择了："+this.blListChoose.Items[e.In-
dex].Text;
    this.lbText.ForeColor=Color.Blue;
}
```

程序运行效果如图 3-19 所示。

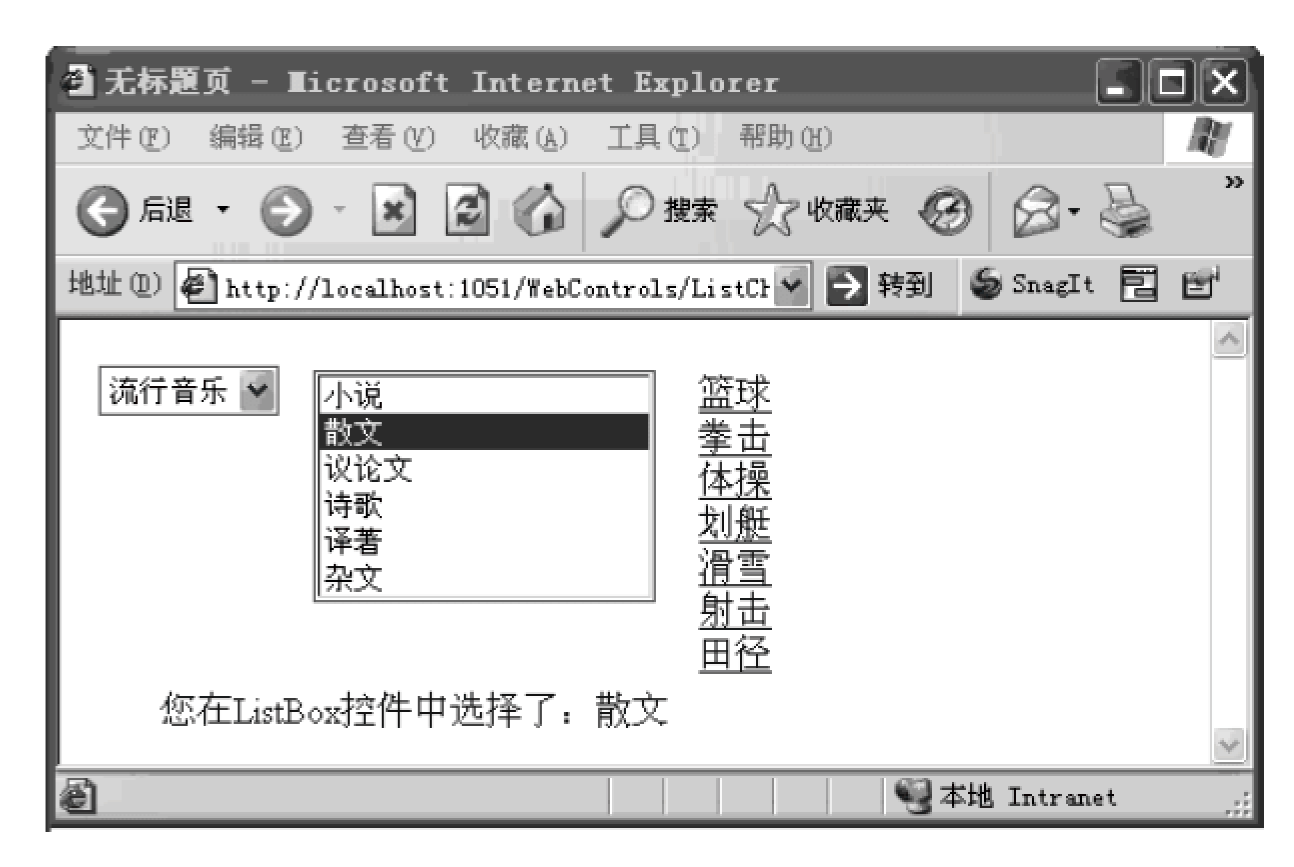


图 3-19　列表控件显示效果图

3.3.2　Web 窗体的数据控件

1. 数据源控件

ASP.NET 包含一些数据源控件，这些数据源控件允许使用不同类型的数据源，如数据库、XML 文件或中间层业务对象。数据源控件连接到数据源，从中检索数据，并使其他控件可以绑定到数据源而无须代码。数据源控件还支持修改数据。

.NET Framework 包含支持不同数据绑定方案的数据源控件。表 3-9 描述了内置的数据源控件。

表 3-9　数据源控件

数据源控件	说　　明
ObjectDataSource	允许使用业务对象或其他类，以及创建依赖中间层对象管理数据的 Web 应用程序。支持对其他数据源控件不可用的高级排序和分页方案
SqlDataSource	允许使用 Microsoft SQL Server、OLE DB、ODBC 或 Oracle 数据库。与 SQL Server 一起使用时支持高速缓存功能。当数据作为 DataSet 对象返回时，此控件还支持排序、筛选和分页
AccessDataSource	允许使用 Microsoft Access 数据库。当数据作为 DataSet 对象返回时，支持排序、筛选和分页
XmlDataSource	允许使用 XML 文件，特别适用于分层的 ASP.NET 服务器控件，如 TreeView 或 Menu 控件。支持使用 XPath 表达式来实现筛选功能，并允许对数据应用 XSLT 转换。XmlDataSource 允许通过保存更改后的整个 XML 文档来更新数据
SiteMapDataSource	结合 ASP.NET 站点导航使用

2. Repeater 控件

Repeater 是一个可重复操作的控件，它通过使用模板显示一个数据源的内容，而且用户可以很容易地配置这些模板。Repeater 控件包含如标题和页脚这样的数据，它可以

遍历所有数据选项并将其应用到模板中。

Repeater 控件并不是由 WebControl 类派生而来的，所以它不包括一些通用的格式属性。但是可以通过直接操纵 HTML 或者 ASP. NET 类来处理这些属性。以下代码简单地声明一个 Repeater 控件。

```
<asp:Repeater id="MyRepeater" runat="server"></asp:Repeater>
```

1）使用模板

在 Web 设计窗口，Repeater 控件显示为一个方框，并有如下提示："切换到 HTML 视图编辑该控件模板"。Repeater 控件只支持模板，它有如下模板可供选择。

- AlternatingItemTemplate：指定如何显示每一个其他选项。
- ItemTemplate：指定如何显示选项。
- HeaderTemplate：建立如何显示标题。
- FooterTemplate：建立如何显示页脚。
- SeparatorTemplate：指定如何显示不同选项之间的分隔符。

用户可以使用这些模板来显示数据。唯一具有强制性的模板是 ItemTemplate，其他模板都是具有可选性的。

2）绑定数据源

对于数据源，Repeater 控件通过属性中的 DataSource 属性获得或者设置 Repeater 显示提供数据源。通过 DataMember 属性获取或者设置与 Repeater 控件绑定相应的 DataSource 属性表格。所有这些操作，也可以通过相应页面中的编程实现。

3）常用事件

Repeater 控件最常用的事件包括 ItemCommand、ItemCreater 和 ItemDataBound。当创建一个项或者一个项被绑定到数据源时，将分别激发 ItemCreated 和 ItemDataBound 事件。当 Repeater 控件中有按钮被单击时，将激发 ItemCommand 事件。在 ItemCommand 事件中，用户可以通过 RepeaterCommandEventArgs 参数获取 CommandArgument、CommandName 和 CommandSource 属性的值。通过这些属性可以获得在 Repeater 中按钮控件的名称、值等。

下面是 Repeater 控件应用实例，程序 3-18 显示使用 Repeater 控件输出图书目录。核心代码如下。

程序 3-18　使用 Repeater 控件输出图书目录

```
<asp:Repeater ID="Repeater1" runat="server" DataSourceID="SqlDataSource1">
        <ItemTemplate>
             • <a  href="Book.aspx?SortID=<%#Eval("SortID") %>">
<%#Eval("SortName")%></a>
                <br />
          </ItemTemplate>
    </asp:Repeater>
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
```

```
ConnectionString="<%$ConnectionStrings:EnglishViewConnectionString %>"
    SelectCommand="SELECT [SortID], [SortName] FROM [BookSort] ORDER BY [SortPRI]">
</asp:SqlDataSource>
```

运行结果如图 3-20 所示。

TUSHUMULU 图书目录

- 剑桥缤纷英语
- 剑桥小学英语
- 超越目标
- 中学英语阅读微技能训练

图 3-20 Repeater 控件显示数据效果图

3. DataList 控件

DataList 控件与 Repeater 控件在很多方面都很相似，但是它也具有一些 Repeater 不具有的属性，特别是它增加了对多列格式、项选择和项编辑的支持。DataList 控件支持各种不同的模板和样式，通过为 DataList 控件的不同部分指定样式，可以自定义该控件的外观。

1）使用模板

在上一部分已经介绍过 Repeater 控件的 5 个模板(ItemTemplate、AlternatingItem-Template、SeparatorTemplate、HeaderTemplate 和 FooterTemplate)，可以格式化控件的输出。在 DataList 控件中除了支持 Repeater 控件中的 5 个模板以外，还支持如下两个模板。

- SelectedItemTemplate：控制如何格式化被选定的项。
- EditItemTemplate：控制如何格式化被编辑的项。

当在 DataList 中选中一个项时(即 DataList 的 SelectedIndex 属性值为当前选定项的索引值)，将显示 SelectedItem 模板；当在 DataList 中选择一个项来编辑(即 DataList 的 EditItemIndex 属性值为当前选定项的索引值)时，将显示 EditItem 模板。

2）在 DataList 中显示数据

可以像 Repeater 控件那样来用 DataList 显示数据库表中的记录。但是，与 Repeater 控件不同的是，DataList 控件的默认行为是在 HTML 表格中显示数据库记录。

3）在 DataList 中创建多列

DataList 的一个好的特征是可以以多个列显示数据项。通过设置其 RepeatColumns 和 RepeatDirection 属性，可以控制 DataList 列的布局。

RepeatColumns 属性决定要显示列的数量。例如，如果要在 DataList 中显示 4 列的项，那么可以把这个属性设为 4。

RepeatDirection 属性是按水平或垂直方向来重复。在默认情况下，RepeatDirection 值为 Vertical，因此，如果 RepeatColumns 值为 4，则列就如下显示。

```
Column 1    Column3    Column5    Column7
Column 2    Column4    Column6    Column8
```

如果把 RepeatDirection 设为 Horizontal，而且 RepeatColumns 值为 4，那么列就如下显示。

```
Column 1    Column3    Column5    Column7
Column 2    Column4    Column6    Column8
```

注意：即使 RepeatDirection 值为 Vertical，还是显示为 4 个列。RepeatColumns 永

远是指重复列的数量，而不是行的数量。

4）DataList 控件支持 5 个事件

- EditCommand：由带有 CommandName＝"edit"的子控件产生。
- CancelCommand：由带有 CommandName＝"cancel"的子控件产生。
- UpdateCommand：由带有 CommandName＝"update"的子控件产生。
- DeleteCommand：由带有 CommandName＝"delete"的子控件产生。
- ItemCommand：DataList 的默认事件。

在 ASP.NET 中有三个控件带有 CommandName 属性，分别是 Button、LinkButton 和 ImageButton，可以设置它们的 CommandName 属性来表示容器控件内产生的时间类型。例如，如果设置 DataList 中的一个 LinkButton 的 CommandName 属性为 update，那么单击此按钮时，将会触发 DataList 的 CancelCommand 事件，我们可以将相关处理代码写到对应的事件处理程序中去。

4. GridView 控件

GridView 控件是功能非常丰富的控件。此控件采用表格形式显示从数据库中获取的数据集合。

1）外观和格式

要定制 GridView 的格式十分简单，只需要用鼠标单击 GridView，在弹出的菜单中选择“自动套用格式”，则可以选择 GridView 的样式。GridView 控件内置了许多样式，如图 3-21 所示。

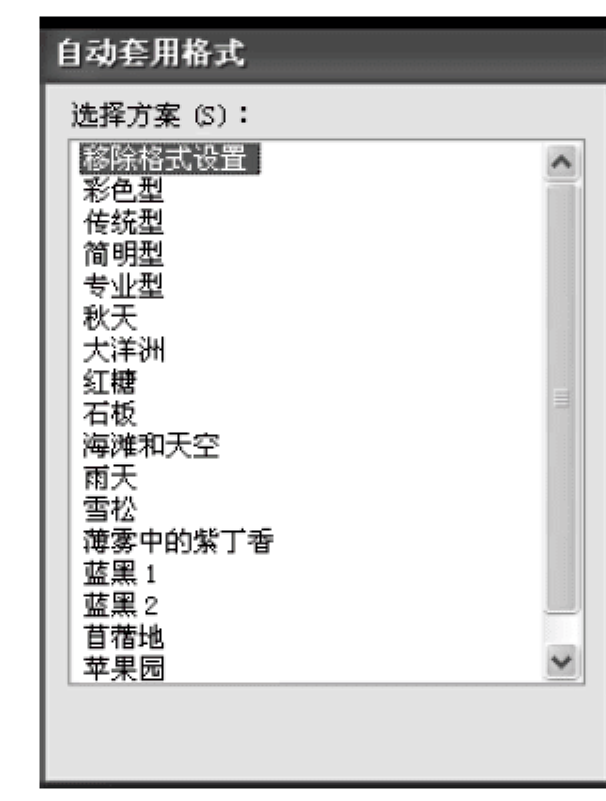


图 3-21　自动套用格式

如果要对 GridView 中每一列自定义格式，则只需单击 GridView 右上角的“智能标记”三角符号，在弹出的菜单中选择“编辑列”，则会弹出如图 3-22 所示的窗体，这样就可以对每列进行详细的设置。例如，添加“编辑”、“更新”和“取消”等按钮。

2）数据绑定

GridView 控件通过数据源控件自动绑定和显示数据。通过数据源控件对数据进行选择、排序、分页、编辑和删除。

GridView 控件提供两个用于数据绑定的选项。

- DataSourceID 属性：此选项能够将 GridView 控件绑定到数据源控件。它允许 GridView 控件利用数据源控件的功能并提供了内置的排序、分页和更新功能。
- DataSource 属性：此选项包括 ADO.NET 数据集和数据读取器在内的各种对象，此方法需要为所有附加功能（如排序、分页和更新）编写代码。

3）常用事件

GridView 控件包含许多事件，其中常用的有 RowDeleting、RowEditing、RowUpdating、RowDeleted 和 RowUpdated 等。通过这些常用属性，可以灵活地实现多种功能。

程序 3-19 是用 GridView 控件对数据进行操作的例子。本实例是通过使用 GridView 对图书进行管理。



图 3-22 自定义格式

程序 3-19 使用 GridView 控件对数据进行显示

```
<asp:GridView ID="GridView1" runat="server" DataSourceID="SqlDataSource1" Width="
700px" AutoGenerateColumns="False" AllowPaging="True">
<Columns>
<asp:BoundField DataField="SortName" HeaderText="分类名称" SortExpression="SortName"
/>
<asp:BoundField DataField="BookPRI" HeaderText="图书优先级" SortExpression="BookPRI"
/>
<asp:BoundField DataField="BookName" HeaderText="书名" SortExpression="BookName" />
<asp:BoundField DataField="BookPublish" HeaderText="出版社" SortExpression="BookPub-
lish" />
<asp:BoundField DataField="BookVariety" HeaderText="品种" SortExpression="BookVari-
ety" />
<asp:BoundField DataField="BookPrice" HeaderText="定价" SortExpression="BookPrice" />
<asp:BoundField DataField="BookSupply" HeaderText="供货" SortExpression="BookSupply"
/>
<asp:HyperLinkField HeaderText="编辑" Text="编辑" DataNavigateUrlFields="BookID,
SortID" DataNavigateUrlFormatString="BookEdit.aspx?BookID={0}&amp;SortID={1}"/>
<asp:HyperLinkField HeaderText="删除" Text="删除" DataNavigateUrlFields="BookID,
SortID" DataNavigateUrlFormatString="BookDel.aspx?BookID={0}&amp;SortID={1}"/>
<asp:HyperLinkField HeaderText="添加" Text="添加" DataNavigateUrlFields="SortID" Da-
taNavigateUrlFormatString="BookAdd.aspx?SortID={0}"/>
</Columns>
</asp:GridView>
```

```
< asp: SqlDataSource  ID =" SqlDataSource1 "  runat =" server "  ConnectionString =" <%
$ConnectionStrings:EnglishViewConnectionString %>"
 SelectCommand="SELECT BookSort.SortName, BookInfo.BookID, BookInfo.SortID, BookInfo.
 BookPRI, BookInfo.BookName,BookInfo.BookVariety, BookInfo.BookPublish, BookInfo.Book-
Price, BookInfo.BookSupply FROM BookInfo INNER JOIN BookSort ON BookInfo.SortID= Book-
Sort.SortID where(BookInfo.[SortID]=@SortID) ORDER BY BookInfo.BookPRI">
    <SelectParameters>
    <asp:QueryStringParameter Name="SortID" QueryStringField="SortID" />
    </SelectParameters>
    </asp:SqlDataSource>
```

运行结果如图 3-23 所示。

图书管理:

分类名称	图书优先级	书名	出版社	品种	定价	供货	编辑	删除	添加
剑桥缤纷英语	1	剑桥缤纷英语 第一级 套装（学生用书+活动手册+音带4盒）	剑桥大学出版社	图书/音带	126.00	全年	编辑	删除	添加
剑桥缤纷英语	2	剑桥缤纷英语 第二级 套装（学生用书+活动手册+音带4盒）	剑桥大学出版社	图书/音带	126.00	全年	编辑	删除	添加
剑桥缤纷英语	3	剑桥缤纷英语 第三级 套装（学生用书+活动手册+音带4盒）	剑桥大学出版社	图书/音带	126.00	全年	编辑	删除	添加
剑桥缤纷英语	4	剑桥缤纷英语 第四级 套装（学生用书+活动手册+音带4盒）	剑桥大学出版社	图书/音带	126.00	全年	编辑	删除	添加
剑桥缤纷英语	5	剑桥缤纷英语 教师用书 第一册 （原版）	剑桥大学出版社	图书	112.00	全年	编辑	删除	添加
剑桥缤纷英语	6	剑桥缤纷英语 教师用书 第二册 （原版）	剑桥大学出版社	图书	112.00	全年	编辑	删除	添加
剑桥缤纷英语	7	剑桥缤纷英语 教师用书 第三册 （原版）	剑桥大学出版社	图书	112.00	全年	编辑	删除	添加
剑桥缤纷英语	8	剑桥缤纷英语 教师用书 第四册 （原版）	剑桥大学出版社	图书	112.00	全年	编辑	删除	添加
剑桥缤纷英语	9	剑桥缤纷英语 教师教学指南 DVD 1盘	剑桥大学出版社	光盘	18.00	全年	编辑	删除	添加
剑桥缤纷英语	10	剑桥缤纷英语 教学卡片1（原版）	剑桥大学出版社	图书	280.00	全年	编辑	删除	添加

1 2

图 3-23　使用 GridView 控件显示数据效果图

3.3.3　验证控件

验证控件是 ASP. NET 引入的一种新型服务器控件，它可以验证服务器控件中用户的输入，并且在验证失败的情况下可以显示一条自定义消息。验证是在客户端直接进行，避免用户提交后，再由服务器返回错误信息的交互过程。ASP. NET 中的验证控件如表 3-10 所示。

表 3-10　6 种验证控件

控　件　名	功 能 描 述
RequiredFieldValidator（必须字段验证）	用于检查是否有输入值
CompareValidator（比较验证）	按设定比较两个输入
RangeValidator（范围验证）	输入是否在指定范围

续表

控 件 名	功 能 描 述
RegularExpressionValidator(正则表达式验证)	正则表达式验证控件
CustomValidator(自定义验证)	自定义验证控件
ValidationSummary(验证总结)	总结验证结果

1. RequiredFieldValidator 控件

控件 RequiredFieldValidator 使用的标准代码如下:

```
<ASP:RequiredFieldValidator id="Validator_Name" Runat="Server"
    ControlToValidate="要检查的控件 ID"
    ErrorMessage="出错信息"
    Display="Static|Dymatic|None"
>
占位符
</ASP: RequiredFieldValidator >
```

在以上标准代码中,ControlToValidate 表示要进行检查控件 ID;ErrorMessage 表示当检查不合法时,出现的错误信息;Display 表示错误信息的显示方式;Static 表示控件的错误信息在页面中占有肯定位置;Dymatic 表示控件错误信息出现时才占用页面控件;None 表示错误出现时不显示,但是可以在 ValidatorSummary 中显示;占位符表示 Display 为 Static 时,错误信息占有占位符那么大的页面空间。

2. CompareValidator 控件

比较两个控件的输入是否符合程序设定,标准代码如下:

```
<ASP:CompareValidator id="Validator_ID" RunAt="Server"
    ControlToValidate="要验证的控件 ID"
    ErrorMessage="错误信息"
    ControlToCompare="要比较的控件 ID"
    Type="String|Integer|Double|DateTime|Currency"
    Operator="Equal|NotEqual|GreaterThan|GreaterThanEqual|LessThan|LessThanEqual|
DataTypeCheck"
    Display="Static|Dymatic|None"
>
占位符
</ASP:CompareValidator>
```

在以上标准代码中,Type 表示要比较的控件的数据类型;Operator 表示比较操作,比较有 7 种方式;其他属性和 RequiredFieldValidator 相同。在这里,要注意 ControlToValidate 和 ControlToCompare 的区别,如果 Operate 为 GreaterThan,那么必须使 ControlToCompare 大于 ControlToValidate 才是合法的。

3. RangeValidator 控件

验证输入是否在一定范围,范围用 MaximumValue(最大)和 MinimumValue(最小)

来确定，标准代码如下：

```
<ASP:RangeValidator id="Validator_ID" Runat="Server"
    ControlToValidate="要验证的控件 ID"
    Type="Integer"
    MinimumValue="最小值"
    MaximumValue="最大值"
    ErrorMessage="错误信息"
    Display="Static|Dymatic|None"
>
占位符
</ASP:RangeValidator>
```

在以上代码中，用 MinimumValue 和 MaximumValue 来界定控件输入值的范围，用 type 来定义控件输入值的类型。

4. RegularExpressionValidator 控件

正则表达式验证控件的功能非常强大，可以自己很容易地构造验证方式。下面来看一下标准代码：

```
<ASP:RegularExpressionValidator id="Validator_ID" Runat="Server"
    ControlToValidate="要验证控件 ID"
    ValidationExpression="正则表达式"
    ErrorMessage="错误信息"
    Display="Static"
>
占位符
</ASP:RegularExpressionValidator>
```

在以上标准代码中，ValidationExpression 不同的字符表示不同的含义："."表示任意字符；"*"表示和其他表达式一起，表示任意组合；"[A-Z]"表示任意大写字母；"\d"表示任意一个数字。

5. ValidationSummary 控件

该控件收集本页的所有验证错误信息，并可以将它们组织以后再显示出来。其标准代码如下：

```
<ASP:ValidationSummary id="Validator_ID" RunaT="Server"
    HeaderText="头信息"
    ShowSummary="True|False"
    DisplayMode="List|BulletList|SingleParagraph"
>
</ASP: ValidationSummary >
```

在以上标准代码中，HeaderText 相当于表的 HeadText，DisplayMode 表示错误信息显示方式：List 相当于 HTML 中的
；BulletList 相当于 HTML 中的<LI>；

SingleParagraph 表示错误信息之间不作分割。

6. CustomValidator 控件

该控件用自定义的函数界定验证方式，其标准代码如下：

```
<ASP:CustomValidator id="Validator_ID" RunAt="Server"
    ControlToValidate="要验证的控件"
    OnServerValidateFunction="验证函数"
    ErrorMessage="错误信息"
    Display="Static|Dymatic|None"
>
占位符
</ASP: CustomValidator>
```

以上代码中，用户必须定义一个函数来验证输入。

3.3.4 导航控件

一个 Web 站点，尤其是信息量大的大中型 Web 站点，应为用户提供站点导航。在网页上，ASP. NET 2.0 提供了站点导航的一种简单方法，即使用站点导航控件 SiteMapPath、TreeView 和 Menu 来完成站点导航。

1. 页面导航的结构

在 ASP. NET 2.0 中，要实现页面导航，应该先以 xml 的形式，提供出整个网站的页面结构层次。可以编写一个叫 web. sitemap 的 XML 文本文件，在该文件中定义出整个要导航页面的结构层次。如程序 3-20 所示。

程序 3-20　XML 文本文件(web. sitemap)

```
<?xml version="1.0" encoding="utf-8" ?>
<sitemap>
<siteMapNode title="Default" description="Home" url="Default.aspx" >
<siteMapNode title="Members" description="Members" url="Members.aspx">
<siteMapNode title="My Account" description="My Account" url="MyAccount.aspx"
/>
<siteMapNode title="Products" description="Products" url="Products.aspx"/>
</siteMapNode>
<siteMapNode title="Administration" description="Administration" url="~/Admin/De-
fault.aspx">
<siteMapNode title="Customer" description="Customer Admin" url="~/Admin/Customer/de-
fault.aspx" />
<siteMapNode title="Products Admin" description="Products Admin"
url="~/Admin/ProductsAdmin.aspx" />
</siteMapNode>
  </siteMapNode>
</sitemap>
```

其中，web.sitemap 文件必须包含根结点 sitemap。而且，设置一个父 sitemapnode 结点，该结点表明是默认的站点首页。在该父 sitemapnode 结点下，可以有若干个子 sitemapnode 结点，分别按层次结构代表了网站的各子栏目(留意一下上例中，各个子结点之间的包含关系)。而每一个 sitemapnode 结点中，有如下若干个属性。

- URL 属性：该属性指出要导航的栏目的地址链接，在 web.sitemap 的定义中，必须是每个栏目的相对地址。
- Title 属性：该属性指出每个子栏目的名称，显示在页面中。
- Description 属性：该属性指定时，用户将鼠标移动到该栏目时，出现有关该栏目的相关提示，类似于 tooltips 属性。

在设计好 sitemap 属性后，接下来就可以一步步构建页面导航功能了，将 sitemap-datasource 控件绑定到如 sitemappath、treeview 和 menu 等控件中。也就是说，将它们的数据源设置为 sitemapdatasource 控件。

2. TreeView 控件

用 Treeview 控件和 sitemapdatasource 控件配合使用的方法。TreeView 树形列表控件十分适合于做页面导航。下面例子具体说明如何使用 TreeView 控件。具体步骤如下所示。

(1) 新建一个网站，添加上文的 web.sitemap 文件。

(2) 添加一个名叫 Navigation 的 master 类型的页面，如程序 3-21 所示。

程序 3-21　导航页面代码

```
<%@ Master Language="C#" %>
<html xmlns="www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
<title>Master Page</title>
</head>
<body>
  <form id="form1" runat="server">
<div>
<table style="width: 100%; height: 100%" border="1">
<tr>
  <td style="width: 10%">
<asp:TreeView ID="TreeView1" Runat="server" DataSourceID="SiteMapDataSource1"
ExpandDepth="2" ShowExpandCollapse="False" NodeIndent="10">
  <LevelStyles>
<asp:TreeNodeStyle Font-Bold="True" Font-Underline="False"/>
<asp:TreeNodeStyle Font-Italic="True" Font-Underline="False" />
<asp:TreeNodeStyle Font-Size="X-Small" ImageUrl="bullet.gif" Font-Underline=
"False" />
  </LevelStyles>
  <NodeStyle ChildNodesPadding="10" />
```

```
</asp:TreeView>
  </td>
  <td style="width: 100px">
<asp:contentplaceholder id="ContentPlaceHolder1" runat="server">
</asp:contentplaceholder>
  </td></tr>
  </table>
  <asp:SiteMapDataSource ID="SiteMapDataSource1" Runat="server"/>
  </div>
</form>
</body>
</html>
```

在上面的代码中，TreeView 控件的 Datasource 属性中，就指定了 sitemapdatasource 控件，并且在 treeview 控件中，也定义了不同结点的样式。在完成了 masterpage 页面后，就等于已经把网站的模板页建立起来了。

（3）接下来就可以新建其他子页面，以继承 masterpage 页面，并且新建各自页面的内容了。新建一个 default.aspx 页面，如程序 3-22 所示。

程序 3-22　设计首页

```
<%@Page Language="C#" MasterPageFile="Navigation.master" Title="Default Page"%>
<asp:Content ContentPlaceHolderID="ContentPlaceHolder1"ID="Content1" Runat="Server"
>
This is the default page
</asp:Content>
```

可以看到，当建立了模板页后，就可以新建其他的子页面了，只需要在其中的 ContentPlaceHolderID 中写入不同的内容就可以了。运行起来后，如图 3-24 所示。

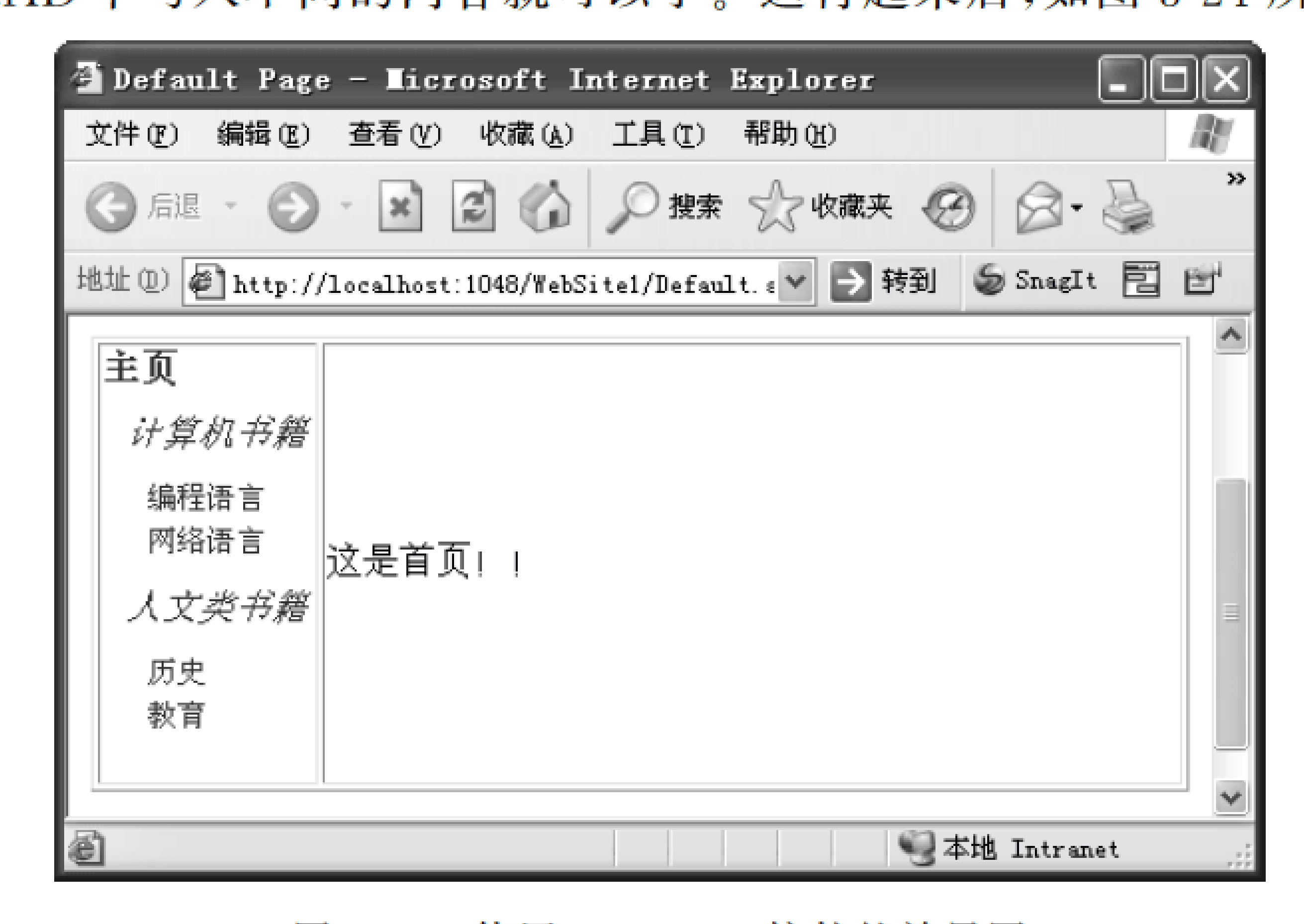


图 3-24　使用 TreeView 控件的效果图

3. Menu 控件

这部分介绍如何将 menu 菜单控件和 sitemapdatasource 控件配合使用。其中，在上面例子的基础上，在＜table style＝"width：100％；height：100％" border＝"1"＞下面增加程序 3-23 所示代码就可以了。

程序 3-23　使用 Menu 控件的代码

```
<tr height="100px">
<td colspan="2" align="left">
<asp:Menu ID="Menu1"  Runat="Server"  DataSourceID="SiteMapDataSource1">
</asp:Menu>
</td>
</tr>
```

其中，增加了一个 menu 控件，将其 datasourceid 属性设定为 sitemapdatasource1 就可以了，运行结果如图 3-25 所示。当然，可以改变 menu 控件的显示位置，如可以将其改成垂直样式显示。



图 3-25　使用 Menu 控件的效果图

4. SiteMapPath 控件

在页面中经常见到的显示出页面当前路径的导航条功能，在 ASP. NET 2.0 中也可以轻易实现，可以使用其中的 sitemappath 控件。紧接着在上文代码中的 menu 控件下，增加程序 3-24 所示代码就可以了。

程序 3-24　使用 SiteMapPath 控件的代码

```
<tr height="100px">
<td colspan="2" align="left">
Currently Selected Page is:
```

```
<asp:SiteMapPath Runat="Server" ID="SiteMapPath1"></asp:SiteMapPath>
</td>
</tr>
```

要注意的是，只要增加 sitemappath 控件就可以了，因为它会自动和已经增加的 sitemapdatasource 控件进行绑定。运行结果如图 3-26 所示。



图 3-26 使用 SiteMapPath 控件的效果图

3.3.5 用户控件和自定义控件

ASP.NET 的组件化并不仅仅局限在使用工具箱中的服务器控件，它还支持用户控件和自定义控件，从而进一步增强了组件的重用和代码共享功能。

1. 用户控件

1）用户控件的基础知识

以下是一个简单的用户控件：

```
<h1>Date:20050101</h1>
```

将这句话保存在以.ascx 为后缀的文件中，就成为用户控件。这个控件非常简单，但是它包含了两条基本原则：用户控件保存在以.ascx 为后缀的文件中；用户控件中没有 <html/><head/><boby/>标记。

用户控件也可以包含方法、属性和事件，只是在这个简单的例子中没有表现。调用这个用户控件，需要使用@Register 指令，如在下面的 ASPX 文件中，就可以调用这个用户控件，如程序 3-25 所示。

程序 3-25　调用用户控件

```
<%@Register TagPrefix="Test" TagName="Date" Src="date.ascx"%>
<html>
<boby>
<form>
  <Test:Date runat="server" id="Counter1"/>
</form>
</boby>
</html>
```

执行这个文件，就可以在页面上以标题一的格式显示 Data：20050101。在调用中，<Test：Date>标记声明了这个用户控件的一个实例。<%@ Register TagPrefix="Test" TagName="Date" Src="date.ascx"%>语句，告知 ASP.NET 在解析到<Test：Date>时如何工作。这其中的关键就是在页面顶部的这个注册说明。

要注册控件，先要定义如下三个属性。

- TagPrefix：定义控件位置的命名空间。有了命名空间制约，就可以在同一页面使用不同功能的同名控件。
- TagName：指向所使用控件的名字。在同一个命名空间里控件名是唯一的。控件名一般都表明控件的功能。
- Src：指向控件资源文件。资源文件使用虚拟路径。

控件注册之后，就可以像其他服务器控件一样被使用。通过定义目标前缀和目标名，即可像使用服务器端内建控件一样地进行使用。同时也确定了使用服务器端运行方式。下面是页面调用用户控件的基本方式：

```
<TagPrefix:TagName runat="server"/>
```

TagPrefix：TagName 结合生成一个标签，当服务器解析时把它与指定的用户控件关联在一起，这个结合对已注册的每个用户控件必须是唯一的。

最简单的放置用户控件的方法是直接从资源管理器中拖放到目的页面上。系统会自动在目的页面上添加需要添加的语句。

2）将 Web 窗体页转换成用户控件

如果用户已经开发了 Web 窗体页，并决定在整个应用程序访问其功能，则可以对该文件进行一些小改动，将其改为用户控件。Web 用户控件与 Web 窗体也非常相似，它们是使用相同的技术创建的。当将 Web 窗体也转换为 Web 用户控件时，创建一个可再次使用的 UI 组件，用户将可以在其他 Web 窗体页上使用该组件。

从 Web 页面到自定义控件需要修改的部分如下。

(1) 删除<html/><head/><boby/>标记。

(2) 把<%@ Page Language="C#" ClassName="*****"%>改为<%@ Control Language="C#" ClassName="*****"%>，把文件存为.ascx 文件即可。

对比 ASCX 文件和 ASPX 文件，它们的异同如表 3-11 所示。

表 3-11　ASCX 文件和 ASPX 文件的异同点

	用 户 控 件	Web 页面
组成文件	一个.ascx 文件和一个.cs 文件	一个.aspx 文件和一个.cs 文件
编写格式	以<%@ Control 开头，其中没有<html>、<form>等标签	以<%@ Page 开头，其中包含<html>、<form>等标签
运行方式	由 CLR 编译，然后存在高速缓存中	相同
能否独立运行	不能，只能被 Web 页调用	可以独立运行

2. 自定义控件

用户自己开发自定义控件并非一件很困难的事。用户只需定义一个直接或间接从 Control 派生的类，并重写它的 Render 方法即可。System. Web. UI. Control 与 System. Web. UI. WEBControls. WebControl 两个类服务器的基类，Control 类定义了所有服务器控件共有的属性、方法和事件，其中包括控制控件执行生命周期的方法和事件，以及 ID、UniqueID、Parent、ViewState 和 Controls（子控件集合）等属性。Control 没有用户界面(UI)特定的任何功能。如果创作的控件没有提供 UI，或者组合了其他呈现其自己的 UI 控件，则从 Control 派生。WebControl 类是从 Control 派生的，并为 UI 功能提供附加的属性和方法。这些属性包括 ForeColor、BackColor、Font、BorderStyle、Height 和 Width。WebControl 是 ASP. NET 中 Web 服务器控件系列的基类。如果控件呈现 UI，则从 WebControl 派生。

用户可以重写从基类继承的属性、方法和事件，并可以将新的属性、方法和事件添加到自定义控件中。如前所述，可以使用重写 Render 方法来实现 HTML 代码的输出。Render 方法采用 System. Web. UI. HtmlTextWriter 类型的参数，控件要发送到客户端的 HTML 作为字符串参数传递到 HtmlTextWriter 的 Write 方法。

3. 用户控件和自定义控件的异同

对用户控件和自定义控件而言，用户控件更适合创建内部的、应用程序特定的控件，而自定义控件则更适合创建通用的和可再分发的控件。具体来讲，可以从 5 个方面说明它们的异同。

(1) 当在两个控件之间进行选择时，部署是最重要的考虑因素。自定义控件采用大多数应用程序可以使用的动态链接库文件(.DLL)为创建可在分发的组件而设计。用户控件是为单个应用程序而设计的，用户控件必须以源代码存在(.ascx 文件)，这种需求有增加维护成本的副作用。

(2) 创建自定义控件的技术与创建用户控件技术是不相同的，自定义控件是用.NET 编程语言通过编写一个从 System. Web. UI. Control 类中直接或者间接派生的托管类而创建，但并没有对创建自定义控件提供设计器支持。而用户控件类间接地从 System. Web. UI. Control 派生而来，用户控件以.ascx 文件的形式声明性地创建，类似于设计和开发 ASP. NET 页面的方式。

(3) 由于创建机制不同，用户控件和自定义控件提供不同的内容和布局方式。自定

义控件非常适用于通过在编程方式生成的布局中表现动态内容。例如，生成一个具有行的数据绑定控件，或一个动态结点的控件树。用户控件的布局在.ascx 文件创建时声明，所以当需要固定布局的相对静态内容时，用户控件是最好的选择。

(4) 目前，Visual Studio.NET 为自定义控件和用户控件提供不同的设计期能力。Visual Studio.Net 为自定义控件提供很大程度的设计期支持功能，例如，可以支持设计时的属性、方法。Visual Studio.NET 对用户控件只有很少的实际期支持。用户控件并不在属性窗口中显示属性和事件，也不能将其放到工具箱中。

(5) 虽然自定义控件和用户控件遵循不同的创建模型，并且有不同的特性，但这两个控件都是从相同的基类直接或间接派生的。用户控件在页面中第一次使用时，将作为普通的服务器控件被解析并编译进配件中。此后用户控件的行为就像任何其他的编译控件一样，所以性能上差别不是很大。

3.4　ASP.NET 基本对象

3.4.1　Application 对象

Application 表示的是应用程序，它是一个可以执行的单一文件，用来完成特定的功能。在 ASP.NET 中 Application 是 ASP.NET 的一个特殊的内置对象，它代表一个目录及所有目录中的 ASP.NET 文件，对于文件之间共享数据是十分方便的。并且，由于它是个内置对象，不需要进行任何对象的实例化，就可以直接使用。

Application 对象的方法和属性如表 3-12 所示。

表 3-12　Application 对象中常见方法和属性

名　　称	方法/属性	说　　明
Lock	方法	锁定 Application 对象，防止其他用户访问 Application 对象
Unlock	方法	撤销对 Application 对象的锁定，其他用户可以访问 Application 对象
Contents 集合	属性	包含所有不使用<OBJECT>标记定义的，已经为会话建立的所有项目
StaticObjects 集合	属性	包含所有使用<OBJECT>标记定义的，在整个应用程序中都有效的所有项目

1. 声明对象

应用程序状态存储在 HttpApplication 类的实例中。在用户初次访问应用程序中的任何 URL 资源时，将会创建 HttpApplicationState 实例。HttpApplicationState 类通常通过 HttpContext 类的 Application 属性访问。

可以在 Web 应用程序的 Global.asax 文件中通过<object runat="server">声明向 StaticObjects 集合添加对象。随后，就可以在应用程序中的任何地方通过代码对用这种方法定义的应用程序状态进行访问。

```
<object runat="server" scope="application" ID="WebInfo" PROGID="MSWC.WEBINFO">
</object>
```

上面的示例代码声明了一个 ID 为 WebInfo 的 Application 对象。

2. 保存 Application 对象中的值

可以通过以下三种方法保存 Application 对象中的值。

(1) 将值写入应用程序状态。将值写入应用程序状态比较简单,示例如下:

```
Application[Message]="欢迎您。";
```

(2) 应用程序启动时将值写入应用程序状态。在应用程序 global.asax 文件的 Application_Start 处理程序中,设置应用程序状态变量的值。同常规的.aspx 页一样,HttpApplicationState 类也通过 Application 对象公开,示例如下:

```
Application[Message]="欢迎您。";
Application[PageReuestCount]=0;
```

(3) 用锁定方法将值写入应用程序状态。应用程序状态变量可以同时被多个线程访问。因此,为了防止产生无效数据,在设置值前,必须锁定应用程序状态,只供一个线程写入,示例如下:

```
Application.Lock();
Application["PageReuestCount"]=((int)Application["PageReuestCount"])+1;
Application.UnLock();
```

3. 从应用程序状态中读取值

应用程序状态存储类型化为 Object 的数据。因此,即使将数据存储于应用程序状态中时不必对其进行序列化,也必须在检查数据时将其强制转换为相应的类型。尽管可以强制转换 null 对象,但是如果尝试通过某些其他方法(如检查其类型)使用不存在的应用程序状态项,就会引发 NullReferenceException 异常。

从应用程序状态中读取值,首先确定应用程序变量是否存在,然后在访问该变量时将其转换成相应的类型,示例如下:

```
if (Application["AppStartTime"]!=null)
{
  DateTime myAppStartTime=(DateTime)Application["AppStartTime"];
}
```

3.4.2 Response 对象

Response 对象是用于把消息向页面上输出的内置对象。在 ASP.NET 的环境中,向页面输出消息的方法有很多,但 Response 对象并不只有一个 Write 方法,它可以提供方法和属性,如表 3-13 所示。

表 3-13 Response 对象常见方法和属性

名 称	方法/属性	说 明
AppHeader	方法	用指定的值给页面添加 HTML 首标信息
AppendToLog	方法	为 Web 服务器添加日志信息
BinaryWrite	方法	将信息不经过字符集转换,直接输出到页面上
Clear	方法	清除缓冲区内容
End	方法	停止当前 ASP. NET 进程的执行
Flush	方法	将缓冲中的内容输出
Redirect	方法	将浏览器重定向到另一个 URL
Write	方法	向页面输出字符串
WriteFile	方法	把文件的内容向页面输出
Buffer	属性	确定数据发送给客户之前是否要缓冲
CacheControl	属性	是否要用高速缓存输出内容
CharSet	属性	字符集名称
ContentType	属性	指定响应的 HTTP 内容的 MIME 类型,默认值是 text/html。也可以用 image/gif 来表示 GIF 图形
Expires	属性	高速缓存过期之前的时间
ExpiresAbsolute	属性	高速缓存过期的日期和时间
IsClientConnected	属性	客户端是否和服务器断开连接
PICS	属性	PICS 标签
Status	属性	返回服务器的状态
Cookies	属性	Cookie 集

1. 重定向

在网页编程中,经常会遇到在程序执行到某个位置进行页面转调的情况。Response. Redirect 方法可以满足这种需求,例如执行代码 Response. Redirect("http://www. 163. com");,页面跳转到网易 163 的主页。

2. 向客户端发送信息

使用 write 方法,向屏幕输出 500 个值。

```
for (int i=1;i<500;i++)
{
 Response.Write("i="+i+"<BR>");
}
```

3. 缓冲 HTML

Buffer 属性可以确定数据发送给客户之前是否要缓冲。它的默认值是 True,表示确

定数据发送给客户之前需要缓冲,也就是说把需要发送给客户的信息先放置到一个预先设定的缓冲区中,然后再一次性地发送给客户。如果把这个值设置为 False,那么所有的信息都会在产生的时候立即发送给客户。从速度上说,这种不使用缓冲的方法会比使用缓冲的方法要慢得多。

3.4.3 Request 对象

Request 对象是用于提取浏览器中用户输入信息的内置对象。在使用这个内部对象时,用户的信息可以通过表单来提供,也可以通过直接用 URL 的参数来提供,还可以通过环境变量来提供。Request 对象的几个常用属性和方法如表 3-14 所示。

表 3-14 Request 对象常用方法和属性

名 称	方法/属性	说 明
TotalBytes	属性	只读属性,它表明用户提交的数据的字节总数
ClientCertication	属性	ClientCertication 集合是客户认证信息集合。如果客户正在使用安全认证,那么从这里可以得到相关的认证信息
Cookies	属性	Cookies 集合。如果使用了 Cookie,那么将从这个集合中读到相关的 Cookie 信息
Form	属性	Form 集合。在发出请求的页面中,所有表单中的控件信息都可以从 Form 集合中得到
QueryString	属性	QueryString 集合。通过页面的 URL 可以得到相关参数的值。这些值可以自动解析并使用
ServerVariables	属性	ServerVariables 集合。通过 ServerVariables 可以得到相关的环境变量。通过获得环境变量的值,可以搜集很多浏览器信息
ApplicationPath	属性	被请求页面的目录信息
Path	属性	被请求页面的完整路径(包括文件名)
PhysicalApplication	属性	被请求页面在硬盘上的物理路径名称
Browser	属性	检索用户浏览器信息
Url	属性	返回浏览器提交的完整的 URL 地址
RawUrl	属性	返回浏览器提交的没有协议和域名的 URL
RequestType	属性	请求 Post 还是 Get
UserHostName	属性	用户的机器名
UserHostAddress	属性	用户机器的 IP 地址
UserLanguages	属性	用户浏览器配置的语言
IsSecureConnection	属性	HTTP 连接是否使用加密
BinaryRead	方法	得到用户上传的文件数总和
MapPath	方法	将当前请求的 URL 中的虚拟路径映射到服务器上的物理路径

1. 获取 QueryString 值

在程序中，经常使用从一个页面传递来的字符串参数。例如，在页面 1 中创建一个连接，指向页面 2，并用 QueryString 来查询两个变量。

```
<a href="page2.aspx?ID=6&Name=Wang">查看</a>
```

在页面 2 中接收到从页面 1 传送过来的两个变量，如下：

```
<Script Language="C#"Runat="Server">
void Page_Load(object sender,System.EventArgs e)
{
 Response.Write("变量 ID 的值："+Request.QueryString["ID"]+"<br>");
 Response.Write("变量 Name 的值："+Request.QuerString["Name"]);
}
</Script>
```

运行上面的代码，结果如下：

```
变量 ID 的值：6
变量 Name 的值：Wang
```

上面的例子可以得到 QueryString 的值。

用类似的方法，可以获得 Form、Cookies 和 ServerVariables 的值。调用方法都是：

```
Request.Collection["Variable"]
```

不过，为了优化程序的执行效率，建议最好还是使用 Collection，因为过多的搜索会降低程序的执行效率。

2. 获取文件的物理路径

通过 Request 对象可以获取文件的物理路径，代码如下：

```
Request.MapPath("FileName");
```

可以通过这条语句来得到某个文件的实际物理位置，这个方法常常用在需要使用实际路径的地方。

3.4.4　Server 对象

Server 对象的属性反映了 Web 服务器的各种信息，它提供了服务器可以提供的各种服务。这个对象不负责为用户存储什么信息，也没有什么事件。

Server 对象常用的方法和属性如表 3-15 所示。

1. 返回服务器计算机名称

通过 Server 对象的 MachineName 属性来获取服务器计算机的名称，示例如下：

表 3-15　Server 对象常用的方法和属性

名　　称	方法/属性	说　　明
MachineName	属性	机器名
ScriptTimeout	属性	说明页面在超时之前可以运行多长时间
CreatObject	方法	创建一个对象
Execute	方法	用于执行一个子例程或存储过程
GetLastError	方法	检索最后一次发生的错误
HtmlDecode	方法	对已被编码的 HTML 字符串进行解码
HtmlEncode	方法	用于把所有的标记字符转换成等价的没有功能的字符
MapPath	方法	得到一个文件的目录路径
Transfer	方法	把原来页面的内置对象的值和对象的状态传递给新的页面
URLEncode	方法	把 URL 中所有的特殊字符转化成对应的非功能等价内容

```
<Script Language="C#"Runat="Server">
void Page_Load(object sender,System.EventArgs e)
{
  String ThisMachine;
  ThisMachine=Server. MachineName;
   Response.Write(ThisMachine);
}
</Script>
```

2. 设置客户端请求的超时期限

用法如下：

```
Server.ScriptTimeout=60;
```

本例中，将客户端请求超时期限设置为 60s，如果 60s 内没有任何操作，服务器将断开与客户的连接。

3. 利用 HtmlEncode 和 HtmlDecode 方法对网页内容编码

当想在网页上显示 HTML 标签时，若在网页中直接输出则会被浏览器解译为 HTML 的内容，所以要通过 Server 对象的 HtmlEncode 方法将它编码再输出。如要将编码后的结果编码回原本的内容，则使用 HtmlDecode 方法。下列程序代码范例使用 HtmlEncode 方法将“<B>HTML 内容</B>”编码后输出至浏览器，再利用 HtmlDecode 方法把编码后的结果编码还原。

```
<Script Language="C# " Runat="Server">
void Page_Load(object sender,System.EventArgs e)
{
```

```
    String strHtmlContent;
    strHtmlContent=Server.HtmlEncode("<B>HTML 内容</B>");
    Response.Write(strHtmlContent);
    Response.Write("<p>");
    strHtmlContent=Server.HtmlDecode("strHtmlContent");
    Response.Write(strHtmlContent);
}
</Script>
```

运行上述示例，通过输出结果可以发现，编码后的 HTML 标注变成了 <B>HTML 内容；</B>，这是因为<B>编程了 <B>，</B>编程了 </B>，所以才能在页面中显示 HTML 标注。

4. 建立虚拟路径与服务器物理目录间映射

使用 MapPath 方法可以将指定相对或虚拟路径映射到服务器上相应的物理目录上。语法如下：

```
public string MapPath(string path);
```

参数 Path 表示指定映射物理目录或虚拟目录路径。若 Path 以一个正斜杠(/)或反斜杠(\)开始，则 MapPath 方法返回路径将 Path 视为完整的虚拟路径。若 Path 不是以斜杠开始，则 MapPath 方法返回同页面文件中已有的路径相对的路径。这里需要注意的是，MapPath 方法不检查返回的路径是否正确或在服务器上是否存在。

对于下列示例，文件 data.txt 和包含下列脚本的 test.aspx 文件都位于目录 c:\Inetpub\wwwroot\aspx 下。c:\Inetpub\wwwroot 目录被设置为服务器的宿主目录。下列示例使用服务器变量 PATH_INFO 映射当前文件的物理路径。以下脚本：

```
Server.MapPath(Request.ServerVariables("PATH_INFO"));
```

输出：

```
C:\interpub\wwwroot\aspx\test.aspx
```

由于下列示例中的路径参数不是以斜杠字符开始的，所以它们被相对映射到当前目录，此处是目录 c:\Inter\wwwroot\aspx。以下脚本：

```
Server.MapPath(data.txt);
Server.MapPath(aspx\data.txt);
```

输出：

```
C:\interpub\wwwroot\aspx\data.txt
C:\interpub\wwwroot\aspx\aspx\data.txt
```

3.4.5 Session 对象

Session 对象是 HttpSessionState 的一个实例。该类为当前用户会话提供信息，还提

供对可用于存储信息的会话范围的缓存访问,以及控制如何管理会话的方法。

Session 对象常用的方法和属性如表 3-16 所示。

表 3-16　Session 对象常用的方法和属性

名　　称	方法/属性	说　　明
Count	属性	获取会话状态集合中 Session 对象的个数
TimeOut	属性	获取并设置在会话状态提供程序终止会话之前,各请求之间允许的超时期限
SessionID	属性	获取用于标识会话的唯一会话 ID
Add	方法	新增一个 Session
Clear	方法	清除会话状态中的所有值
Remove	方法	删除会话状态集合中的项
RemoveAll	方法	消除所有会话状态值

1. 获取 Session 对象的个数

Count 属性可以帮助统计正在使用的 Session 对象的个数,语句非常简单,示例如下:

```
Respose.Write(Session.Count);
```

2. 通过 Add 方法设置 Session 对象

通过 Add 方法可以设置 Session 对象的值,语法如下:

```
Session.Add("变量名",变量值);
```

在具体应用中,可以这样使用:

```
int userId=1;
string userName="test";
string userPwd="admin";
Session.Add("userId",userId);
Session.Add("userName", userName);
Session.Add("userPwd", userPwd);
```

在上面的例子中创建了 userId、userName 和 userPwd 三个 Session 来存储用户的登录信息。程序随时都可以通过这三个 Session 对象来查看用户的连接状态,这是实际的项目中最常见的 Session 应用。

需要注意的是,也可以不使用 Add 方法来设置 Session 对象,语法如下:

```
Session["变量名"]=变量值;
```

这样,上面的例子就可以改为:

```
Session["userId"]=userId;
Session["userName"]=userName;
```

```
Session["userPwd"]=userPwd;
```

需要指出的是，以上两种语句在作用上是相同的，可以根据自己的习惯来使用。

3.4.6　Cookie 对象

Cookie 是一小段文本信息，伴随着用户请求和页面在 Web 服务器和浏览器之间传递。用户每次访问站点时，Web 应用程序都可以读取 Cookie 包含的信息。Cookie 跟 Session、Application 类似，也是用来保存相关信息。但 Cookie 和其他对象的最大不同是，Cookie 将信息保存在客户端，而 Session 和 Application 是保存在服务器端。也就是说，无论何时用户连接到服务器，Web 站点都可以访问 Cookie 信息。这样，既方便用户的使用，也方便了网站对用户的管理。

Cookie 对象常用的方法和属性如表 3-17 所示。

表 3-17　Cookie 对象常用的方法和属性

名　称	方法/属性	说　明
Name	属性	获取或设置 Cookie 的名称
Value	属性	获取或设置 Cookie 的 Value
Expires	属性	获取或设置 Cookie 的过期日期和时间
Version	属性	获取或设置此 Cookie 符合的 HTTP 状态维护版本
Add	方法	新增一个 Cookie 变量
Clear	方法	清除 Cookie 集合内的变量
Get	方法	通过变量名或索引得到 Cookie 的变量值
GetKey	方法	以索引来获取 Cookie 的变量名称
Remove	方法	通过 Cookie 变量名来删除 Cookie 变量

1. 设置 Cookie

下面的示例将创建名为 LastVisit 的新 Cookie，将该 Cookie 的值设置为当前日期和时间，并将其添加到当前 Cookie 集合中，所有 Cookie 均通过 HTTP 输出流在 Set-Cookie 头中发送到客户端。

```
HttpCookie MyCookie=new HttpCookie("LastVisit");
DateTime now=DateTime.Now;
MyCookie.Value=now.ToString();
MyCookie.Expires=now.AddHours(1);
Response.Cookie.Add(MyCookie);
```

运行上面的例子，将会在用户机器的 Cookies 目录建立如下内容的文本文件。

```
Mycookie
LastVisit
```

尽管上面的这个例子很简单，但可以从中扩展许多富有创造性的应用程序。

2. 获取客户端发送的 Cookie 信息

下面的示例是依次通过客户端的所有 Cookie，并将每个 Cookie 的名称、过期日期、安全参数和值发送到 HTTP 输出，如程序 3-26 所示。

程序 3-26　获取客户端发送的 Cookie 信息

```
int loop1,loop2;
HttpCookieCollection MyCookieColl;
HttpCookie MyCookie;
MyCookieColl=Request.Cookies;
//把所有的 Cookie 名放到一个字符数组中
String[] arr1=MyCookieColl.AllKeys;
//用 Cookie 名获取单个 Cookie 对象
for(loop1=0;loop1<arr1.Length;loop1++)
{
 MyCookie=MyCookieColl[arr1[loop1]];
 Response.Write("Cookie:"+MyCookie.Name+"<br>");
 Response.Write("Expires:"+MyCookie. Expires+"<br>");
Response.Write("Secure:"+MyCookie. Secure+"<br>");
//将单个 Cookie 的值放入一个对象数组
String [] arr=MyCookie.Values>allKeys;
//遍历 Cookie 值集合打印所有值
for(loop2=0;loop2<arr2.Length;Loop2++)
{
 Respones.Write("Value"+loop2+":"+arr2[loop2]+"<br>");
}
}
```

3.4.7　Cache 对象

对于每个应用程序域均创建该类的一个示例，并且只要对应的应用程序域保持活动，该示例保持有效。有关此类示例的信息通过 HttpContext 对象的 Cache 属性或 Page 对象的 Cache 属性来提供。

Cache 对象常用的方法和属性如表 3-18 所示。

表 3-18　Cache 对象常用的方法和属性

名　　称	方法/属性	说　　明
Count	属性	获取存储在缓存中的项数。当监视应用程序性能或使用 ASP. NET 跟踪功能时，此属性可能非常有用
Item	属性	获取或设置指定键外的缓存项
Add	方法	将指定项添加到 Cache 对象

续表

名 称	方法/属性	说 明
Get	方法	从 Cache 对象检索指定项
Remove	方法	从应用程序的 Cache 对象移除指定项
Insert	方法	向 Cache 对象插入项。使用此方法的某一个版本改写具有相同 key 参数的现有 Cache 项

1. 检索为 ASP.NET 文本框服务器控件缓存的值

Get 方法可以从 Cache 对象检索指定项，其唯一的参数 key 表示要检索的缓存项的标识符。该方法返回检索到的缓存项，未找到该键时为空引用。

下面的示例展示如何检索为 ASP.NET 文本框服务器控件缓存的值。

```
Cache.Get("MyTextBox.Value");
```

2. 移除 Cache 对象

Remove 方法可以 Cache 对象移除指定项，其唯一的参数 key 表示要移除的缓存项的 String 标识符，该方法返回 Cache 移除项。如果未找到键参数中的值，则返回空引用。

第 4 章　数据库基础与 ADO. NET

现在的网站一般都是动态页面，支持用户交互，代码与数据库分离。很多网站一般都是数据库服务器和 Web 服务器分离。故我们构建网站时离不开与数据库的交互。在 . NET 中，与数据库的交互是通过 ADO. NET 来实现的。

ADO. NET 是对 Microsoft ActiveX Data Objects (ADO)一个跨时代的改进，它提供了平台互用性和可伸缩的数据访问。由于传送的数据都是 XML 格式的，因此任何能够读取 XML 格式的应用程序都可以进行数据处理。事实上，接受数据的组件不一定是 ADO. NET 组件，它可以是基于一个 Microsoft Visual Studio 的解决方案，也可以是任何运行在其他平台上的应用程序。

ADO. NET 是一组用于和数据源进行交互的面向对象类库。通常情况下，数据源是数据库，但它同样也能够是文本文件、Excel 表格或者 XML 文件。因为本章的需要，我们将 ADO. NET 当作是一种与数据库的交互方式。

ADO. NET 允许和不同类型的数据源及数据库进行交互，然而并没有与此相关的一系列类来完成这样的工作。因为不同的数据源采用不同的协议，所以对于不同的数据源必须采用相应的协议。一些老式的数据源使用 ODBC 协议，许多新的数据源使用 OleDb 协议，并且现在还在不断出现更多的数据源，这些数据源都可以通过 . NET 的 ADO . NET 类库来进行连接。

ADO. NET 提供与数据源进行交互的相关的公共方法，但是对于不同的数据源采用一组不同的类库。这些类库称为 Data Providers，并且通常是以与之交互的协议和数据源的类型来命名的。

ADO. NET 是与数据源交互的 . NET 技术。有许多的 Data Providers，它将允许与不同的数据源交流——取决于它们所使用的协议或者数据库。然而无论使用什么样的 Data Provider，将使用相似的对象与数据源进行交互。SqlConnection 对象管理与数据源的连接。SqlCommand 对象允许与数据源交流并发送命令给它。为了进行快速的只“向前”地读取数据，使用 SqlDataReader。如果想使用断开数据，使用 DataSet 并实现能进行读取或者写入数据源的 SqlDataAdapter。

4.1　数据库基础

4.1.1　数据库定义和分类

1. 数据库定义

数据库是一个以某种有组织的方式存储的数据集合。为了使一个数据库可以真正具有功能性，它不仅要储存大量的记录良好，更要进入容易。此外，新的信息和变化，也应该是相当容易的投入。为了有一个高效的数据库系统，需要把这种程式管理的质疑和资料储存于系统。这通常被人们称为数据库管理系统。

2. 数据库的分类

数据库中的数据是有结构的，这些结构反映了事物之间的联系，对这种结构的描述就是数据模型，是表示数据与数据之间联系的方法。不同的数据模型以不同的方式把数据组织到数据库中，常用的数据模型有三种：层次模型、关系模型和网络模型。

1）层次模型数据库

层次模型以树形结构表示实体(记录)与实体之间的联系。层次模型像一棵倒置的树，根结点在上，层次最高，子结点在下，逐层排列。

层次模型的特点：有且仅有一个结点无父结点，这个结点即为树的根；其他结点有且仅有一个结点。

2）网络模型

网络模型是以网状表示实体与实体之间的联系。网状模型可以表示多个从属关系的联系，也可以表示数据间的交叉关系，即数据间的横向关系与纵向关系，它是层次模型的扩展。

网状模型的特点：可以有一个以上的结点无父结点；至少有一个子结点有一个以上的父结点；在两个结点之间有两个或两个以上的联系。

3）关系模型

关系模型是目前最流行的类型的数据库和一个极其有力的工具。关系模型是把数据结构看成一个二维表，每个二维表就是一个关系，关系模型是由若干个二维表格组成的集合，是以关系数学理论为基础的。在关系模型中，操作的对象和结果都是二维表，这种二维表就是关系。

在二维表中，每一行称为一个记录，用于表示一组数据项；表中的每一列称为一个字段或属性，用于表示每一列中的数据项；表中的第一行称为字段名，用于表示每个字段的名称。

关系模型的特点：表格中的每一列都是不可再分的基本属性；每一列都被指定一个不相同的名字；各行不允许重复；行、列的次序无关。

关系型数据库的使用程序界面是所谓的 SQL 或标准查询语言。SQL 是目前所采用的几乎所有的关系型数据库。关系数据库产品是非常容易定制的，以适应几乎任何类型

的数据存储。

4.1.2 SQL Server 数据库简介

美国 Microsoft 公司推出的一种关系型数据库系统。SQL Server 是一个可扩展的、高性能的、为分布式客户机/服务器计算所设计的数据库管理系统,实现了与 Windows NT 的有机结合,提供了基于事务的企业级信息管理系统方案。其主要特点如下。

- 高性能设计,可充分利用 Windows NT 的优势。
- 系统管理先进,支持 Windows 图形化管理工具,支持本地和远程的系统管理和配置。
- 强壮的事务处理功能,采用各种方法保证数据的完整性。
- 支持对称多处理器结构、存储过程和 ODBC,并具有自主的 SQL 语言。SQL Server 以其内置的数据复制功能、强大的管理工具、与 Internet 的紧密集成和开放的系统结构,为广大的用户、开发人员和系统集成商提供了一个出众的数据库平台。

本章的研究主要是基于 SQL Server 数据库。

1. 数据库的主码(主键)

能够唯一表示数据表中每个记录的“字段”或者“字段”的组合就称为主码。

2. 数据库表

数据表是数据库中一个非常重要的对象,是其他对象的基础。没有数据表,关键字、主键和索引等也就无从谈起。在数据库画板中可以显示数据库中的所有数据表(即使不是用 PowerBuilder 创建的表)、创建数据表和修改表的定义等。

关系数据库通常包含多个表。数据库实际上是表的集合,数据库的数据或者信息都是存储在表中的。表是对数据进行存储和操作的一种逻辑结构,每一个表都代表一个对用户意义的对象。例如,一个公司数据库中,有雇员表、部门表、库存表、销售表和工资表等。我们经常见到的成绩表就是一种表,它是由行和列组成的,并且可以通过名字来识别数据。列包含了列的名字、数据类型及列的其他属性;行包含了列的记录或者数据。下面给出一个成绩单,如表 4-1 所示,其中姓名、语文、数学和英语都是列,而行包含了这个表的数据,即每个人的各科成绩。

表 4-1 成绩单

姓　名	语　文	数　学	英　语
王小童	78	100	87
张柳风	85	92	95
紫云飞	65	89	86
黄天龙	98	67	75

3. 数据操作语言

DML(Data Manipulation Language,数据操作语言)用于检索或者修改数据。DML

组可以细分为以下几个语句。

- SELECT：用于检索数据。
- INSERT：用于增加数据到数据库。
- UPDATE：用于从数据库中修改现存的数据。
- DELETE：用于从数据库中删除数据。

4. 数据检索

在 SQL 中 SELECT 语句通常用于检索数据库，或者检索满足设定条件的数据，以下是简单的 SELECT 语句的格式：

```
select "column1"[,"column2",etc] from "tablename" [where "condition"];
[]=optional
```

其中列的名字跟着 SELECT 关键字，它决定了哪一列将被作为结果返回。可以任意指定多个列，或者可以使用“*”来选择所有的列。

表的名字是紧跟着 FROM 关键字的，它指出了哪个表格将作为最后结果被查询。而 WHERE 子句(可选)指出哪个数据或者行将被返回或者显示，它是根据关键字 WHERE 后面描述的条件而来的。

在 WHERE 子句中可以有以下的条件选择：=(等于)、>(大于)、<(小于)、>=(大于等于)、<=(小于等于)和<>(不等于)。LIKE 参见以下注释。

注释：LIKE 模式匹配操作符同样可以使用在 WHERE 子句的条件中。LIKE 是一个功能强大的操作符，它可以选择“喜欢”指定的行。“%”可以被用来匹配任何可能的字符，它可以出现在指定字符的前面或者后面，例如：

```
select first, last, city  from empinfo  where first LIKE 'Er% ';
```

以上这条 SQL 语句将会匹配任何名字以 Er 开始的名字，这里必须使用单引号。或者也可以使用%在字符的前面，例如：

```
select first, last from empinfo where last LIKE '% s';
```

这条 SQL 语句将会匹配任何名字以 s 结尾的名字。这个%的作用就跟 DOS 命令的“*”号很相似。

```
select * from empinfo where first='Eric';
```

以上的 SQL 语句只选择 first 名字为 Eric 的行。

5. 插入数据到表

Insert 语句用于往表格中插入或者增加一行数据，它的格式为：

```
insert into "tablename"(first_column,…,last_column)values (first_value,…,last_value);
[]=optional
```

简单举个例子：

```
insert into employee(first, last, age, address, city) values ('Luke', 'Duke', 45, '2130
Boars Nest', 'Hazard Co');
```

这里要注意，每一个字符串都要用单引号括起来。

为了往表中插入数据，要在关键字 insert into 之后紧跟着表名，然后是左圆括号，接着是以逗号分开的一系列的列名，再是一个右圆括号，然后在关键字 values 之后跟着一系列用圆括号括起的数值。这些数值是要往表格中填入的数据，它们必须与指定的列名相匹配。字符串必须用单引号括起来，而数字不用。在上面的例子中，'Luke'必须与列 first 相匹配，而 45 必须与列 age 相匹配。

假如想往 employee 表格中插入以下数据：

```
Zhang Weiguo,28,北京 601 信箱,北京
```

那么要使用以下的 SQL 语句：

```
insert into employee (first, last, age, address, city)  values (' Zhang', ' Weiguo',28, '
北京 601 信箱', '北京');
```

6. 删除记录

Delete 语句是用来从表中删除记录或者行，其语句格式为：

```
delete from "tablename"
where "columnname" OPERATOR "value" [and|or "column" OPERATOR "value"];
[ ]=optional
```

下面还是举个例子：

```
delete from employee;
```

这条语句没有 where 语句，所以它将删除所有的记录，因此如果没有使用 where 的时候，要千万小心。如果只要删除其中一行或者几行，可以参考以下的语句：

```
delete from employee where lastname= 'May';
```

这条语句是从 employee 表中删除 lastname 为'May'的行。

```
delete from employee where firstname= 'Mike' or firstname= 'Eric';
```

这条语句是从 employee 表中删除 firstname 为'Mike'或者'Eric'的行。为了从表中删除一个完整的记录或者行，就直接在"delete from"后面加上表的名字，并且利用 where 指明符合什么条件的行要删除即可。如果没有使用 where 子句，那么表中的所有记录或者

行将被删除。

7. 更新记录

Update 语句用于更新或者改变匹配指定条件的记录，它是通过构造一个 where 语句来实现的。其语句格式如下：

```
update "tablename" set "columnname"="newvalue"[,"nextcolumn"="newvalue2"…]
where "columnname" OPERATOR "value" [and|or "column" OPERATOR "value"];
[]=optional
```

下面举个例子来说明：

```
update phone_book  set area_code=623 where prefix=979;
```

以上语句是在 phone_book 表中，在 prefix=979 的行中将 area_code 设置为 623。

```
update phone_book set last_name='Smith', prefix=555, suffix=9292 where last_name=
'Jones';
```

而以上的这段语句是在 phone_book 中，在 last_name='Jones'的行中将 last_name 设置为'Smith'，prefix 为 555，suffix 为 9292。

```
update employee set age=age+1 where first_name='Mary' and last_name='Williams';
```

这段语句是在 employee 表中，在 first_name='Mary'和 last_name='Williams'的行中将 age 加 1。

8. 存储过程

SQL 语句执行时要先编译，然后执行。存储过程就是编译好了的一些 SQL 语句。应用程序需要用的时候直接调用就可以了，所以效率很高。

存储过程是由流控制和 SQL 语句书写的过程，这个过程经编译和优化后存储在数据库服务器中，应用程序使用时只要调用即可。在 Oracle 中，若干个有联系的过程可以组合在一起构成程序包。

使用存储过程有以下的优点。

(1) 存储过程的能力大大增强了 SQL 语言的功能和灵活性。存储过程可以用流控制语句编写，有很强的灵活性，可以完成复杂的判断和较复杂的运算。

(2) 可保证数据的安全性和完整性。

(3) 通过存储过程可以使没有权限的用户在控制之下间接地存取数据库，从而保证数据的安全。

(4) 通过存储过程可以使相关的动作在一起发生，从而可以维护数据库的完整性。

(5) 在运行存储过程前，数据库已对其进行了语法和句法分析，并给出了优化执行方案。这种已经编译好的过程可极大地改善 SQL 语句的性能。由于执行 SQL 语句的大部

分工作已经完成，所以存储过程能以极快的速度执行。

(6) 可以降低网络的通信量。

(7) 将体现企业规则的运算程序放入数据库服务器中，以便：

① 集中控制。

② 当企业规则发生变化时在服务器中改变存储过程即可，无须修改任何应用程序。企业规则的特点是要经常变化，如果把体现企业规则的运算程序放入应用程序中，则当企业规则发生变化时，需要修改应用程序的工作量非常大(修改、发行和安装应用程序)。如果把体现企业规则的运算放入存储过程中，则当企业规则发生变化时，只要修改存储过程就可以了，应用程序无须任何变化。

数据库存储过程的实质就是部署在数据库端的一组定义代码及 SQL。

4.2 ADO.NET 模型简介

ADO.NET 提供对 Microsoft SQL Server 等数据源及通过 OLE DB 和 XML 公开的数据源的一致访问。数据共享使用者应用程序可以使用 ADO.NET 来连接到这些数据源，并检索、操作和更新数据。

ADO.NET 有效地从数据操作中将数据访问分解为多个可以单独使用或一前一后使用的不连续组件。ADO.NET 包含用于连接到数据库、执行命令和检索结果的.NET Framework 数据提供程序。可以直接处理检索到的结果，或将其放入 ADO.NET DataSet 对象，以便与来自多个源的数据或在层之间进行远程处理的数据组合在一起，以特殊方式向用户公开。ADO.NET DataSet 对象也可以独立于.NET Framework 数据提供程序使用，以管理应用程序本地的数据或源自 XML 的数据。

ADO.NET 类在 System.Data.dll 中，并且与 System.Xml.dll 中的 XML 类集成。当编译使用 System.Data 命名空间的代码时，引用 System.Data.dll 和 System.Xml.dll。

以前，数据处理主要依赖于基于连接的双层模型。当数据处理越来越多地使用多层结构时，程序员正在向断开方式转换，以便为他们的应用程序提供更佳的可缩放性。

1. XML 和 ADO.NET

ADO.NET 借用 XML 的力量来提供对数据的断开式访问。ADO.NET 的设计与.NET Framework 中 XML 类的设计是并进的，它们都是同一个结构的组件。ADO.NET 和.NET Framework 中的 XML 类集中于 DataSet 对象。无论 DataSet 是文件还是 XML 流，它都可以使用来自 XML 源的数据来进行填充。无论 DataSet 中数据的数据源是什么，DataSet 都可以写为符合万维网联合会的 XML，DataSet 包含了 XML Schema 语言(XML Schema Definition，XSD)架构。由于 DataSet 固有的序列化格式为 XML，它是在层间移动数据的优良媒介，这使 DataSet 成为以远程方式向 XML Web Services 发送数据和架构上下文，以及从 XML Web Services 接收数据和架构上下文的最佳选择。DataSet 也可与 XmlDataDocument 进行同步，以实时地提供对数据的关系和分层访问。

2. ADO.NET 组件

设计 ADO.NET 组件是为了从数据操作中分解出数据访问。ADO.NET 的两个核

心组件会完成此任务：DataSet 和.NET Framework 数据提供程序，后者是一组包括 Connection、Command、DataReader 和 DataAdapter 对象在内的组件。

ADO.NET DataSet 是 ADO.NET 的断开式结构的核心组件。DataSet 的设计目的很明确：实现独立于任何数据源的数据访问。因此，它可以用于多种不同的数据源，用于 XML 数据，或用于管理应用程序本地的数据。DataSet 包含一个或多个 DataTable 对象的集合，这些对象由数据行和数据列，以及主键、外键、约束和有关 DataTable 对象中数据的关系信息组成。

ADO.NET 结构的另一个核心元素是.NET Framework 数据提供程序，其组件的设计目的相当明确：实现数据操作和对数据的快速、只进、只读访问，Connection 对象提供与数据源的连接。Command 对象使用户能够访问用于返回数据、修改数据、运行存储过程，以及发送或检索参数信息的数据库命令。DataReader 从数据源中提供高性能的数据流。最后，DataAdapter 提供连接 DataSet 对象和数据源的桥梁。DataAdapter 使用 Command 对象在数据源中执行 SQL 命令，以便将数据加载到 DataSet 中，并使对 DataSet 中数据的更改与数据源保持一致。可以为任何数据源编写.NET Framework 数据提供程序。

.NET Framework 提供了 4 个.NET Framework 数据提供程序：SQL Server .NET Framework 数据提供程序、OLE DB .NET Framework 数据提供程序、ODBC .NET Framework 数据提供程序和 Oracle .NET Framework 数据提供程序。为了使应用程序获得最佳性能，应该使用适合的数据源的.NET 数据提供程序。

例如，如果要访问 SQL Server 2000 数据库，则应该使用 SQL Server .NET Framework 数据提供程序。而如果是要访问 Access 数据库，则应该使用 OLE DB .NET Framework 数据提供程序。

图 4-1 阐释了 ADO.NET 组件的结构模型。

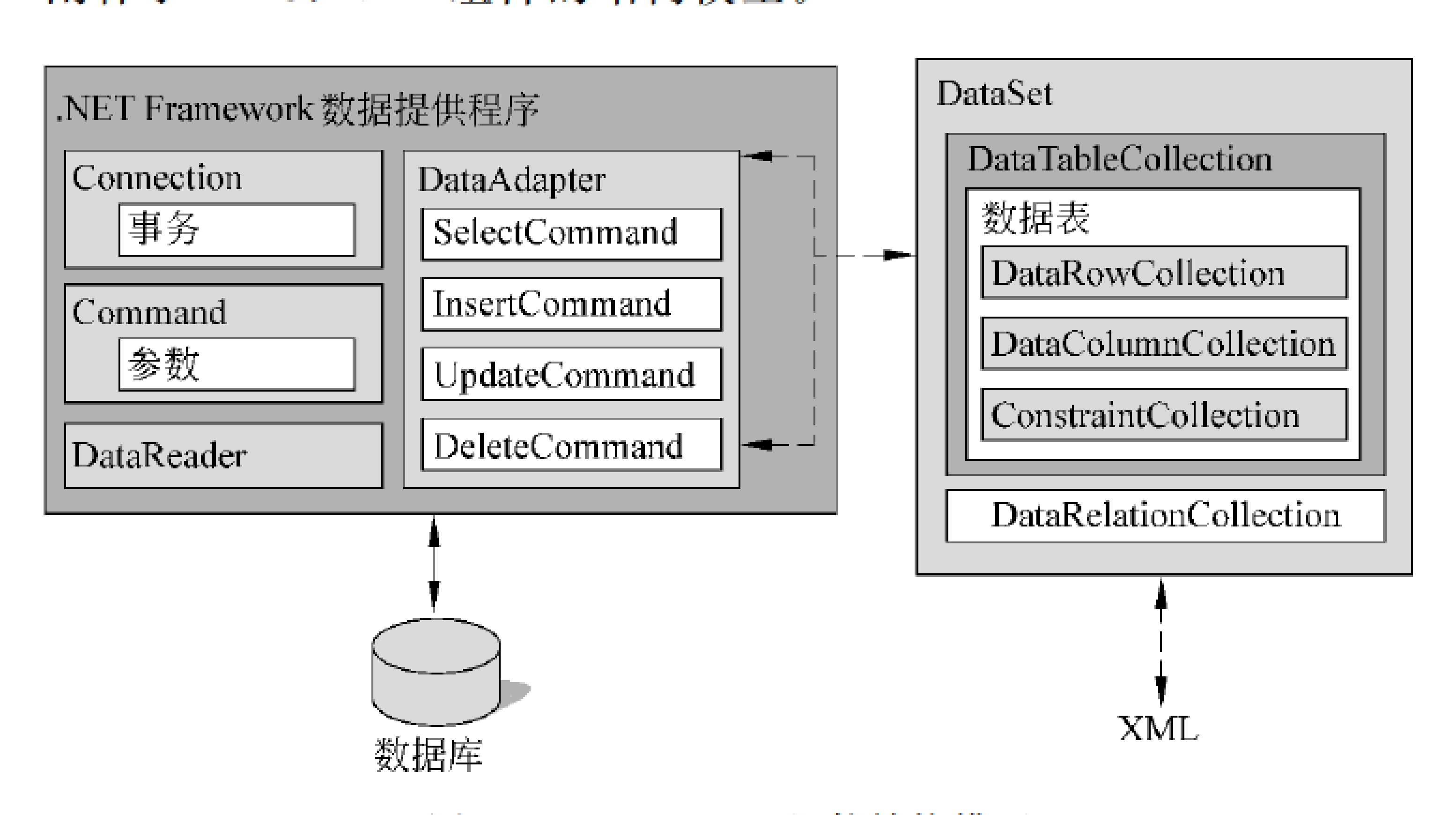


图 4-1　ADO.NET 组件结构模型

DataSet 的设计使用户能够使用 XML Web Services 方便地通过 Web 将数据传输到客户端，并允许使用 .NET 远程处理服务在 .NET 组件之间传送数据。还可以通过这种方式对强类型的 DataSet 进行远程处理。DataTable 对象也可以与远程处理服务一起使

用，但不能通过 XML Web Services 进行传输。

3. ADO.NET 平台要求

支持 Microsoft.NET Framework SDK（包括 ADO.NET）的平台包括 Microsoft Windows 2000、Windows NT 4（带 Service Pack 6a）、Windows Millennium Edition、Windows 98 和 Windows SE。使用 SQL Server.NET Framework 数据提供程序或 OLE DB .NET Framework 数据提供程序要求安装 Microsoft 数据访问组件 2.6 版或更高版本。

以下代码示例显示如何将 System.Data 命名空间包含在用户的应用程序中，以使用 ADO.NET。

```
[C#]
using System.Data;
```

ADO.NET 类在 System.Data.dll 中，并且与 System.Xml.dll 中的 XML 类集成。当编译使用 System.Data 命名空间的代码时，引用 System.Data.dll 和 System.Xml.dll。

4.3 ADO.NET 组件

Connection、Command、DataReader 和 DataAdapter 对象是.NET Framework 数据提供程序模型的核心元素。表 4-2 将描述这些对象。

表 4-2 数据提供程序模型的核心元素

对　　象	说　　明
Connection	建立与特定数据源的连接
Command	对数据源执行命令
DataReader	从数据源中读取只进且只读的数据流
DataAdapter	用数据源填充 DataSet 并解析更新

4.3.1 数据连接

1. 使用 ADO.NET 连接到数据源

在数据访问中首先必须建立到数据库的物理连接。在 ADO.NET 中，可以使用 Connection 对象标识与一个数据库的物理连接，但实际上 ADO.NET 中并没有一个名字为 Connection 的类用来创建 Connection 对象。事实上，每个数据提供程序都包含自己特有的 Connection 对象。若要连接到 Microsoft SQL Server 7.0 版或更高版本，应该使用 SQL Server .NET Framework 数据提供程序的 SqlConnection 对象。若要使用用于 SQL Server 的 OLE DB 提供程序（SQLOLEDB）连接到 OLE DB 数据源或者连接到 Microsoft SQL Server 6.x 版或较早版本，要使用 OLE DB .NET Framework 数据提供程序

的 OleDbConnection 对象。若要连接到 ODBC 数据源，则使用 ODBC .NET Framework 数据提供程序的 OdbcConnection 对象。若要连接到 Oracle 数据源，则使用 Oracle .NET Framework 数据提供程序的 OracleConnection 对象。

2. 使用 ADO.NET 连接 SQL Server 数据源

SQL Server .NET Framework 数据提供程序使用 SqlConnection 对象提供与 Microsoft SQL Server 7.0 版或更高版本的连接。其中 SqlConnection 对象是在 System.Data.SqlClient 命名空间中定义的，它包含在 System.Data.DLL 程序集中。使用 Visual Studio.NET 开发应用程序时，一般项目都自动引用了该程序集。可以使用如下两种方式来使用 SqlConnection 类。

```
[C#]
System.Data.SqlClient. SqlConnection conn=new System.Data.SqlClient. SqlConnection();
```

下面是在类中导入了这个命名空间后的使用方式。

```
[C#]
using System.Data.SqlClient;
SqlConnection conn=new SqlConnection();
```

显然，实际编码时采用后一种方式会减少代码编写量。

1）SqlConnection 类

为了连接 SQL Server，必须实例化 SqlConnection 对象，并调用此对象的 Open 方法。当不再需要连接时，应该调用这个对象的 Close()方法关闭连接。SqlConnection 类定义了两个构造函数，一个不带参数，另外一个接受连接字符串。因此可以使用下面两种方式的任何一种实例化连接。

```
SqlConnection conn=new SqlConnection();
conn.ConnectionString=ConnectionString;
conn.Open();
```

或者：

```
SqlConnection conn=new SqlConnection(ConnectionString);
conn.Open();
```

显然，使用后一种方法输入的代码要少一点，但是两种方法执行的效率并没有什么不同。另外，如果需要重用 Connection 对象去使用不同的身份连接不同的数据库时，使用第一种方法则非常有效。例如：

```
SqlConnection conn=new SqlConnection();
conn.ConnectionString=ConnectionString1;
```

```
conn.Open();
//访问数据库,做一些事情
conn.Close();

conn.ConnectionString=ConnectionString2;
conn.Open();
//访问数据库,做另外一些事情
conn.Close();
```

但是必须注意,只要当一个连接关闭以后才能把另外一个不同的连接字符串赋值给Connection 对象。如果不知道 Connection 对象在某个时候是打开还是关闭,可以检查Connection 对象的 State 属性,它的值可以是 Open,也可以是 Closed,这样就可以知道连接是否是打开的。

2) SQL Server 的连接字符串

SQL Server .NET Framework 数据提供程序支持类似于 OLE DB(ADO)连接字符串格式的连接字符串格式。连接字符串包含一个由一些"属性名/值"对组成的集合。每一个"属性名/值"对都有分号隔开。如:

```
PropertyName1=Value1; PropertyName2=Value2;…;PropertyNameN=ValueN;
```

以下代码示例演示如何创建和打开与 SQL Server(版本 7.0 或更高版本)数据库的连接。

```
[C#]
using System.Data.SqlClient;
SqlConnection conn=new SqlConnection();
conn.ConnectionString="server=(local); Initial Catalog=northwind; User Id=sa; Password
=admin");
conn.Open();
```

其中,server 为服务器地址,其值(local)表示使用本地机器,另外可以使用计算机名作为服务器的值。Initial Catalog 属性指明了连接使用的数据库,上面的连接字符串表明使用数据库 northwind。User Id 和 Password 则分别指明了访问数据库时使用的用户名和密码。

3) SqlConnection 的方法

SqlConnection 对象有自己的方法,如前面已经使用过的 Open 方法和 Close 方法。表 4-3 列出了其常用的方法。

数据库连接是很有价值的资源,因为连接要使用到宝贵的系统资源,如内存和网络带宽,因此对数据库的连接必须小心使用,要在最晚的时候建立连接(调用 Open 方法),在最早的时候关闭连接(调用 Close 方法)。也就是说,在开发应用程序时,不再需要数据连接时应该立刻关闭数据。

表 4-3　SqlConnection 对象的方法

方 法 名 称	方 法 描 述
Open	使用 ConnectionString 所指定的属性设置打开数据库连接
Close	关闭与数据库的连接,这是关闭任何打开连接的首选方法
CreateCommand	创建并返回一个与 SqlConnection 关联的 SqlCommand 对象
ChangeDatabase	为打开的 SqlConnection 更改当前数据库

3. 集成安全性和 ASP.NET

SQL Server 集成安全性(也称为受信任的连接)在连接到 SQL Server 时可以提供保护,因为它不会在连接字符串中公开用户标识和密码。建议使用该方法对连接进行身份验证。集成安全性使用正在执行的进程的当前安全标识或标记。对于桌面应用程序,安全标识或标记通常是当前登录的用户的标识。ASP.NET 应用程序的安全标识可设置为几个不同的选项之一。

下面的示例创建一个 SqlCommand 和一个 SqlConnection,SqlCommand 将在下面介绍。SqlConnection 打开,并设置为 SqlCommand 的 Connection。然后,该示例调用 ExecuteNonQuery 并关闭该连接。为了完成此任务,将为 ExecuteNonQuery 传递一个连接字符串和一个查询字符串,后者是一个 Transact-SQL INSERT 语句。

```
[C#]
public void InsertRow(string myConnectionString)
 {
    //如果 myConnectionString 为空,则赋一个默认值
    if(myConnectionString=="")
    {
     myConnectionString="Initial Catalog=Northwind;Data Source=localhost;Integrated
Security=SSPI;";
    }
    SqlConnection myConnection=new SqlConnection(myConnectionString);
     string myInsertQuery="INSERT INTO Customers (CustomerID, CompanyName) Values
('NWIND','Northwind Traders')";
    SqlCommand myCommand=new SqlCommand(myInsertQuery);
    myCommand.Connection=myConnection;
    myConnection.Open();
    myCommand.ExecuteNonQuery();
    myCommand.Connection.Close();
}
```

4.3.2 数据命令

1. 执行命令

当建立了数据连接之后，就可以执行数据访问和数据操纵了。一般对数据库的操作被概括为 CRUD(Create、Read、Update 和 Delete)。可以使用 Command 构造函数来创建命令，该构造函数采用在数据源、Connection 对象和 Transaction 对象中执行的 SQL 语句的可选参数。也可以使用 Connection 的 CreateCommand 方法来创建用于特定 Connection 对象的命令。可以使用 CommandText 属性来查询和修改 Command 对象的 SQL 语句。Command 对象公开了几个可用于执行所需操作的 Execute 方法。当以数据流的形式返回结果时，使用 ExecuteReader 可返回 DataReader 对象。使用 ExecuteScalar 可返回单个值，使用 ExecuteNonQuery 可执行不返回行的命令。

当将 Command 对象用于存储过程时，可以将 Command 对象的 CommandType 属性设置为 StoredProcedure。当 CommandType 为 StoredProcedure 时，可以使用 Command 的 Parameters 属性来访问输入及输出参数和返回值。无论调用哪一个 Execute 方法，都可以访问 Parameters 属性。但是，当调用 ExecuteReader 时，在 DataReader 关闭之前，将无法访问返回值和输出参数。

以下代码示例演示如何设置 Command 对象的格式，以便从 Northwind 数据库中返回 Categories 的列表。

```
[C#]
string commandText="SELECT CategoryID, CategoryName FROM Categories";
SqlConnection conn=new SqlConnection();
conn.ConnectionString="server=(local); Initial Catalog=northwind; User Id=sa; Password
=admin");
SqlCommand cmd=new SqlCommand(commandText,conn);
```

或者：

```
SqlConnection conn=new SqlConnection();
conn.ConnectionString="server=(local); Initial Catalog=northwind; User Id=sa; Password
=admin");
SqlCommand catCMD=new SqlCommand("SELECT CategoryID, CategoryName FROM Categories",
conn);
```

2. 将存储过程用于命令

存储过程提供了数据驱动应用程序中的许多优点。利用存储过程，数据库操作可以封装在单个命令中，为获取最佳性能而进行优化并通过附加的安全性得到增强。虽然可以通过以 SQL 语句的形式传递参数自变量之前的存储过程名称来调用存储过程，但如果使用 ADO.NET Command 对象的 Parameters 集合，则可以显式地定义存储过程参数并

访问输出参数和返回值。

若要调用存储过程，则将 Command 对象的 CommandType 设置为 StoredProcedure。CommandType 一旦设置为 StoredProcedure，就可以使用 Parameters 集合来定义参数，如以下示例所示。

注释：LIKE 模式匹配操作符同样可以使用在 WHERE 子句的条件中。LIKE 是一个功能强大的操作符，它可以让用户选择“喜欢”指定的行。%可以被用来匹配任何可能的字符，它可以出现在指定字符的前面或者后面。

注意：OdbcCommand 要求在调用存储过程时提供完整的 ODBC CALL 语法。

```
[C#]
SqlConnection nwindConn=new SqlConnection("Data Source=localhost;Integrated Security=
SSPI;Initial Catalog=northwind");
SqlCommand salesCMD=new SqlCommand("SalesByCategory", nwindConn);
salesCMD.CommandType=CommandType.StoredProcedure;
SqlParameter myParm=salesCMD.Parameters.Add("@CategoryName", SqlDbType.NVarChar, 15);
myParm.Value="Beverages";
nwindConn.Open();
SqlDataReader myReader=salesCMD.ExecuteReader();
Console.WriteLine("{0}, {1}", myReader.GetName(0), myReader.GetName(1));
while (myReader.Read())
{
  Console.WriteLine("{0}, ${1}", myReader.GetString(0), myReader.GetDecimal(1));
}
myReader.Close();
nwindConn.Close();
```

Parameter 对象可以使用 Parameter 构造函数来创建，或通过调用 Command 的 Parameters 集合的 Add 方法来创建。Parameters. Add 会将构造函数参数或现有 Parameter 对象用作输入。在将 Parameter 的 Value 设置为空引用时，使用 DBNull. Value。

对于 Input 参数之外的参数，必须设置 ParameterDirection 属性来指定参数类型是 Input、Output 还是 ReturnValue。以下示例显示创建 Input、Output 和 ReturnValue 参数之间的差异。

```
[C#]
SqlCommand sampleCMD=new SqlCommand("SampleProc", nwindConn);
sampleCMD.CommandType=CommandType.StoredProcedure;
SqlParameter sampParm=sampleCMD.Parameters.Add("RETURN_VALUE", SqlDbType.Int);
sampParm.Direction=ParameterDirection.ReturnValue;
sampParm=sampleCMD.Parameters.Add("@InputParm", SqlDbType.NVarChar, 12);
sampParm.Value="Sample Value";
```

```
sampParm=sampleCMD.Parameters.Add("@OutputParm", SqlDbType.NVarChar, 28);
sampParm.Direction=ParameterDirection.Output;

nwindConn.Open();

SqlDataReader sampReader=sampleCMD.ExecuteReader();

Console.WriteLine("{0}, {1}", sampReader.GetName(0), sampReader.GetName(1));

while (sampReader.Read())
{
  Console.WriteLine("{0}, {1}", sampReader.GetInt32(0), sampReader.GetString(1));
}

sampReader.Close();
nwindConn.Close();

Console.WriteLine(" @OutputParm: {0}", sampleCMD.Parameters["@OutputParm"].Value);
Console.WriteLine("RETURN_VALUE: {0}", sampleCMD.Parameters["RETURN_VALUE"].Value);
```

3. 将参数用于 SqlCommand

在将参数用于 SqlCommand 时，添加到 Parameters 集合的参数的名称必须与存储过程中参数标记的名称相匹配。SQL Server .NET Framework 数据提供程序将存储过程中的参数视为命名参数并搜索匹配的参数标记。SQL Server .NET Framework 数据提供程序不支持在向 SQL 语句或存储过程传递参数时使用问号(?)占位符。在这种情况下，必须使用命名参数，如以下示例所示。

```
SELECT * FROM Customers WHERE CustomerID=@CustomerID
```

4. 导出参数信息

参数也可以使用 CommandBuilder 类从存储过程导出。SqlCommandBuilder 和 OleDbCommandBuilder 类都提供了静态方法 DeriveParameters，该静态方法将自动使用存储过程中的参数信息填充 Command 对象的 Parameters 集合。请注意，DeriveParameters 将改写 Command 的任何现有参数信息。

导出参数信息时需要经历一个到数据源的附加行程，以获取参数信息。如果参数信息在设计时是已知的，则可以通过显式设置参数来提高应用程序的性能。

以下代码示例显示如何使用 CommandBuilder. DeriveParameters 来填充 Command 对象的 Parameters 集合。

```
 [C#]
SqlConnection nwindConn=new SqlConnection("Data Source=localhost;Initial
```

```
Catalog=Northwind;Integrated Security=SSPI;");
SqlCommand salesCMD=new SqlCommand("Sales By Year", nwindConn);
salesCMD.CommandType=CommandType.StoredProcedure;
nwindConn.Open();
SqlCommandBuilder.DeriveParameters(salesCMD);
nwindConn.Close();
```

5. 从数据库中获取单个值

用户可能需要返回只是单个值的数据库信息，而不需要返回表或数据流形式的数据库信息。例如，可能需要返回 Count(＊)、Sum(Price)或 Avg(Quantity) 等聚合函数的结果。Command 对象使用 ExecuteScalar 方法提供了返回单个值的功能。ExecuteScalar 方法以标量值的形式返回结果集第一行中第一列的值。

以下代码示例使用 Count 聚合函数来返回表中记录的数目。

```
[C#]
SqlCommand ordersCMD=new SqlCommand("SELECT Count(*) FROM Orders", nwindConn);
Int32 count=(Int32)ordersCMD.ExecuteScalar();
```

6. 执行数据库操作和修改数据

使用.NET Framework 数据提供程序，可以执行存储过程或数据定义语言(Data Definition Language，DDL)语句(如 CREATE TABLE 和 ALTER COLUMN)来对数据库或目录执行架构操作。这些命令不会像查询一样返回行，因此 Command 对象提供了 ExecuteNonQuery 方法来处理这些命令。

除了使用 ExecuteNonQuery 来修改架构之外，还可以使用此方法处理那些修改数据但不返回行的 SQL 语句，如 INSERT、UPDATE 和 DELETE 等。

虽然行不是由 ExecuteNonQuery 方法返回的，但可以通过 Command 对象的 Parameters 集合来传递和返回输入及输出参数和返回值。

7. 执行目录操作

若要执行对数据库或目录进行修改的命令(如 CREATE TABLE 或 CREATE PROCEDURE 语句)，使用相应的 Transact-SQL 语句和 Connection 创建一个 Command。使用 Command 对象的 ExecuteNonQuery 方法来执行该命令。

以下代码示例在 Microsoft SQL Server 数据库中创建一个存储过程。

```
[C#]
string createStr="CREATE PROCEDURE InsertCategory  "+
                           "  @CategoryName nchar(15), "+
                           "  @Identity int OUT "+
                           "AS "+
```

```
                    "INSERT INTO Categories (CategoryName) VALUES (@Catego-
                    ryName) "+
                  "SET @Identity=@@Identity "+
                  "RETURN @@ROWCOUNT";
SqlCommand createCMD=new SqlCommand(createStr, nwindConn);
createCMD.ExecuteNonQuery();
```

8. 修改数据库中的数据

修改数据的 SQL 语句(如 INSERT、UPDATE 或 DELETE)不返回行。同样,许多存储过程执行操作但不返回行。若要执行不返回行的命令,使用相应的 SQL 命令和 Connection(以及任何必需的 Parameters)创建一个 Command 对象,然后使用该 Command 对象的 ExecuteNonQuery 方法。ExecuteNonQuery 方法返回一个整数,表示受已执行的语句或存储过程影响的行数。如果执行了多个语句,则返回的值为受所有已执行语句影响的记录的总数。

以下代码示例执行一个 INSERT 语句,以使用 ExecuteNonQuery 将一个记录插入数据库中。

```
 [C#]
SqlConnection nwindConn=new SqlConnection("Data Source=localhost;Integrated Security=
SSPI; Initial Catalog=northwind");
nwindConn.Open();

string insertStr="INSERT INTO Customers (CustomerID, CompanyName) Values('NWIND', '
Northwind Traders')";
SqlCommand insertCMD=new SqlCommand(insertStr, nwindConn);
Int32 recordsAffected=insertCMD.ExecuteNonQuery();
```

4.3.3 数据集

当执行返回结果集的命令时,需要一个方法从结果集中提取数据。处理结果集的方法有如下两个。

① 使用数据阅读器(DataReader)。

② 同时使用数据适配器(DataAdapter)和 ADO.NET 数据集(DataSet)。

当确定应用程序应使用 DataReader 还是应使用 DataSet 时,应考虑应用程序所需的功能类型。DataSet 用于执行以下功能。

(1) 在层间或从 XML Web Services 对数据进行远程处理。

(2) 与数据进行动态交互,例如绑定到 Windows 窗体控件或组合并关联来自多个源的数据。

(3) 在应用程序本地缓存数据。

(4) 提供关系数据的分层 XML 视图并使用 XSL 转换或 XML 路径语言(XPath)查询等工具来处理数据。

(5) 对数据执行大量的处理，而不需要与数据源保持打开的连接，从而将该连接释放给其他客户端使用。

如果不需要 DataSet 所提供的功能，则可以使用 DataReader 以只进只读方式返回数据，从而提高应用程序的性能。虽然 DataAdapter 使用 DataReader 来填充 DataSet 的内容，但可以使用 DataReader 来提高性能，因为这样将节省 DataSet 所使用的内存，并将省去创建 DataSet 并填充其内容所需的必要处理。

本节主要讲解使用数据适配器和 ADO. NET 数据集读取数据。下一节讲述数据阅读器。

1. 从 DataAdapter 填充 DataSet

ADO. NET DataSet 是数据的内存驻留表示形式，它提供了独立于数据源的一致关系编程模型。DataSet 表示整个数据集，其中包含表、约束和表之间的关系。由于 DataSet 独立于数据源，DataSet 可以包含应用程序本地的数据，也可以包含来自多个数据源的数据。与现有数据源的交互通过 DataAdapter 来控制。. NET Framework 所包含的每个. NET Framework 数据提供程序都具有一个 DataAdapter 对象：OLE DB. NET Framework 数据提供程序包含 OleDbDataAdapter 对象，SQL Server. NET Framework 数据提供程序包含 SqlDataAdapter 对象，ODBC . NET Framework 数据提供程序包含 OdbcDataAdapter 对象。DataAdapter 对象用于从数据源中检索数据并填充 DataSet 中的表。DataAdapter 还会将对 DataSet 作出的更改解析回数据源。DataAdapter 使用 . NET Framework 数据提供程序的 Connection 对象连接到数据源，使用 Command 对象从数据源中检索数据并将更改解析回数据源。

DataAdapter 的 SelectCommand 属性是一个 Command 对象，它从数据源中检索数据。DataAdapter 的 InsertCommand、UpdateCommand 和 DeleteCommand 属性也是 Command 对象，它们按照对 DataSet 中数据的修改来管理对数据源中数据的更新。

DataAdapter 的 Fill 方法用于使用 DataAdapter 的 SelectCommand 的结果来填充 DataSet。Fill 将要填充的 DataSet 和 DataTable 对象(或要使用从 SelectCommand 中返回的行来填充 DataTable 的名称)用作它的参数。

Fill 方法使用 DataReader 对象来隐式地返回用于在 DataSet 中创建表的列名称和类型，以及用来填充 DataSet 中的表行的数据。表和列仅在不存在时才创建；否则，Fill 将使用现有的 DataSet 架构。列类型按照将. NET Framework 数据提供程序数据类型映射到. NET Framework 数据类型中的表创建为. NET Framework 类型。除非数据源中存在主键并且 DataAdapter. MissingSchemaAction 设置为 MissingSchemaAction. AddWithKey，否则不会创建主键。如果 Fill 发现存在用于某表的主键，那么对于其中的主键列值与从数据源中返回的行的主键值相匹配的行，它将使用数据源中的数据改写 DataSet 中的数据。如果未找到任何主键，则数据将追加到 DataSet 中的表。当填充

DataSet 时,Fill 会使用任何可能存在的 TableMappings。

注意:如果 SelectCommand 返回 OUTER JOIN 的结果,则 DataAdapter 不会为结果 DataTable 设置 PrimaryKey 值。需要自己定义 PrimaryKey 以确保正确解析重复行。

以下代码示例创建 DataAdapter 的一个实例,该实例使用 Microsoft SQL Server Northwind 数据库的 Connection 并使用客户列表来填充 DataSet 中的 DataTable。向 DataAdapter 构造函数传递的 SQL 语句和 Connection 参数用于创建 DataAdapter 的 SelectCommand 属性。

```
[C#]
SqlConnection nwindConn=new SqlConnection("Data Source=localhost;Integrated Security=
SSPI;Initial Catalog=northwind");

SqlCommand selectCMD=new SqlCommand("SELECT CustomerID, CompanyName FROM Customers",
nwindConn);
selectCMD.CommandTimeout=30;

SqlDataAdapter custDA=new SqlDataAdapter();
custDA.SelectCommand=selectCMD;

nwindConn.Open();

DataSet custDS=new DataSet();
custDA.Fill(custDS, "Customers");

nwindConn.Close();
```

注意:该代码不显式打开和关闭 Connection。如果 Fill 方法发现连接尚未打开,它将隐式地打开 DataAdapter 正在使用的 Connection。如果 Fill 已打开连接,它还将在 Fill 完成时关闭连接。当处理单一操作(如 Fill 或 Update)时,这可以简化用户的代码。但是,如果在执行多项需要打开连接的操作,则可以通过以下方式提高应用程序的性能:显式调用 Connection 的 Open 方法,对数据源执行操作,然后调用 Connection 的 Close 方法。为了释放资源供其他客户端应用程序使用,应设法使与数据源的连接打开尽可能短的时间。

2. 多个结果集

如果 DataAdapter 遇到多个结果集,它将在 DataSet 中创建多个表。将向这些表提供递增的默认名称 TableN,以表示 Table0 的 Table 为第一个表名。如果以参数形式向 Fill 方法传递表名称,则将向这些表提供递增的默认名称 TableNameN,这些表名称以表示 TableName0 的 TableName 为起始。

3. 从多个 DataAdapter 填充 DataSet

可以将任意数量的 DataAdapter 与一个 DataSet 一起使用。每个 DataAdapter 都可

用于填充一个或多个 DataTable 对象并将更新解析回相关数据源。DataRelation 和 Constraint 对象可以在本地添加到 DataSet，这样，就可以使来自多个不同数据源的数据相关联。例如，DataSet 可以包含来自 Microsoft SQL Server 数据库、通过 OLE DB 公开的 IBM DB2 数据库及对 XML 进行流处理的数据源的数据。一个或多个 DataAdapter 对象可以处理与每个数据源的通信。

以下代码示例从 Microsoft SQL Server 2000 上的 Northwind 数据库填充客户列表，从存储在 Microsoft Access 2000 中的 Northwind 数据库填充订单列表。已填充的表通过 DataRelation 相关联，然后客户列表将与相应客户的订单一起显示出来。

```
[C#]
SqlConnection custConn=new SqlConnection("Data Source=localhost;Integrated Security=
SSPI;Initial Catalog=northwind;");
SqlDataAdapter custDA=new SqlDataAdapter("SELECT * FROM Customers", custConn);

OleDbConnection orderConn=new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;"+
                                              "Data Source=c:\\Program Files\\Mi-
                                              crosoft Office\\Office\\Samples\\
                                              northwind.mdb;");
OleDbDataAdapter orderDA=new OleDbDataAdapter("SELECT * FROM Orders", orderConn);

custConn.Open();
orderConn.Open();

DataSet custDS=new DataSet();

custDA.Fill(custDS, "Customers");
orderDA.Fill(custDS, "Orders");

custConn.Close();
orderConn.Close();

DataRelation custOrderRel=custDS.Relations.Add("CustOrders",
                              custDS.Tables["Customers"].Columns["CustomerID"],
                              custDS.Tables["Orders"].Columns["CustomerID"]);
foreach (DataRow pRow in custDS.Tables["Customers"].Rows)
{
  Console.WriteLine(pRow["CustomerID"]);
   foreach (DataRow cRow in pRow.GetChildRows(custOrderRel))
    Console.WriteLine("\t"+cRow["OrderID"]);
}
```

4. 使用 DataAdapter 和 DataSet 更新数据库

DataAdapter 的 Update 方法可调用来将 DataSet 中的更改解析回数据源。与 Fill 方法类似，Update 方法将 DataSet 的实例和可选的 DataTable 对象或 DataTable 名称用作参数。DataSet 实例是包含已作出更改的 DataSet，而 DataTable 标识从其中检索更改的表。

当调用 Update 方法时，DataAdapter 将分析已作出的更改并执行相应的命令（INSERT、UPDATE 或 DELETE）。当 DataAdapter 遇到对 DataRow 的更改时，它将使用 InsertCommand、UpdateCommand 或 DeleteCommand 来处理该更改。这样，就可以通过在设计时指定命令语法，并在可能时通过使用存储过程来尽量提高 ADO.NET 应用程序的性能。在调用 Update 之前，必须显式设置这些命令。如果调用了 Update 但不存在用于特定更新的相应命令（例如，不存在用于已删除行的 DeleteCommand），则将引发异常。

Command 参数可用于为 DataSet 中每个已修改行的 SQL 语句或存储过程指定输入和输出值。

如果 DataTable 映射到单个数据库表或从单个数据库表生成，则可以利用 CommandBuilder 对象自动生成 DataAdapter 的 DeleteCommand、InsertCommand 和 UpdateCommand。

Update 方法会将更改解析回数据源，但是自上次填充 DataSet 以来，其他客户端可能已修改了数据源中的数据。若要使用当前数据刷新 DataSet，则再次使用 DataAdapter 填充（Fill）DataSet。新行将添加到该表中，更新的信息将并入现有行。Fill 方法通过检查 DataSet 中行的主键值及 SelectCommand 返回的行来确定是否要添加一个新行或更新现有行。如果 Fill 方法发现 DataSet 中某行的主键值与 SelectCommand 返回结果中某行的主键值相匹配，则它将用 SelectCommand 返回的行中的信息更新现有行，并将现有行的 RowState 设置为 Unchanged。如果 SelectCommand 返回的行所具有的主键值与 DataSet 中行的任何主键值都不匹配，则 Fill 方法将添加 RowState 为 Unchanged 的新行。

注意：如果 SelectCommand 返回 OUTER JOIN 的结果，则 DataAdapter 不会为结果 DataTable 设置 PrimaryKey 值。需要自己定义 PrimaryKey 以确保正确解析重复行。

若要处理可能在 Update 操作过程中发生的异常，可以使用 RowUpdated 事件在这些异常发生时响应行更新错误，或者可以在调用 Update 之前将 DataAdapter.ContinueUpdateOnError 设置为 true，然后在 Update 完成时响应存储在特定行的 RowError 属性中的错误信息。

注意：如果对 DataSet、DataTable 或 DataRow 调用 AcceptChanges，则将使一个 DataRow 的所有 Original 值都被改写为该 DataRow 的 Current 值。如果已修改将该行标识为唯一行的字段值，那么当调用 AcceptChanges 后，Original 值将不再匹配数据源中的值。

以下示例演示如何通过显式设置 DataAdapter 的 UpdateCommand 来执行对已修改行的更新。请注意，在 UPDATE 语句的 WHERE 子句中指定的参数设置为使用 SourceColumn 的 Original 值。这一点很重要，因为 Current 值可能已被修改，并且可能

不匹配数据源中的值。Original 值是曾用来从数据源填充 DataTable 的值。

```
[C#]
SqlDataAdapter catDA=new SqlDataAdapter("SELECT CategoryID, CategoryName FROM Catego-
ries",nwindConn);
catDA.UpdateCommand=new SqlCommand("UPDATE Categories SET CategoryName=@CategoryName "
+"WHERE CategoryID=@CategoryID", nwindConn);
catDA.UpdateCommand.Parameters.Add("@CategoryName", SqlDbType.NVarChar, 15, "Catego-
ryName");
SqlParameter workParm= catDA.UpdateCommand.Parameters.Add("@CategoryID", SqlDbType.
Int);
workParm.SourceColumn="CategoryID";
workParm.SourceVersion=DataRowVersion.Original;
DataSet catDS=new DataSet();
catDA.Fill(catDS, "Categories");
DataRow cRow=catDS.Tables["Categories"].Rows[0];
cRow["CategoryName"]="New Category";
catDA.Update(catDS);
```

5. 自动递增列

如果来自数据源的表包含自动递增列，则可以使用由数据源生成的值填充 DataSet 中的列，方法是通过以存储过程输出参数的形式返回自动递增值并将其映射到表中的一列，或者使用 DataAdapter 的 RowUpdated 事件。

但是，DataSet 中的值可能会与数据源中的值不同步并导致意外的行为。例如，请考虑一个包含自动递增主键列 CustomerID 的表。如果在该 DataSet 中添加两个新客户，它们将收到自动递增的 CustomerId 值 1 和 2。在向 DataAdapter 的 Update 方法传递第二个客户行时，新添加的行会收到数据源中的自动递增 CustomerID 值 1，该值与 DataSet 中的值 2 不匹配。当 DataAdapter 使用返回值填充 DataSet 中的行时，由于第一个客户行的 CustomerID 已经是 1，因此将发生约束冲突。为了避免这种行为，建议在使用数据源中的自动递增列和 DataSet 中的自动递增列时，在 DataSet 中创建 AutoIncrementStep 为－1 且 AutoIncrementSeed 为 0 的列，并确保数据源生成从 1 开始并以正步长值递增的自动递增标识值。这样，DataSet 将为自动递增值生成负数，这些负数不会与数据源所生成的正自动递增值发生冲突。另一种方法是使用 Guid 类型的列而不是自动递增列。生成 Guid 值的算法在 DataSet 中生成的 Guid 不会与数据源生成的 Guid 相同。

6. 插入、更新和删除的排序

在许多情况下，以何种顺序向数据源发送通过 DataSet 作出的更改是相当重要的。例如，如果已更新现有行的主键值并且添加了具有新主键值的新行，则务必要在处理插入之前处理更新。可以使用 DataTable 的 Select 方法来返回仅引用具有特定 RowState 的 DataRow 数组。然后可以将返回的 DataRow 数组传递到 DataAdapter 的 Update 方法来

处理已修改的行。通过指定要更新的行的子集，可以控制处理插入、更新和删除的顺序。

例如，以下代码确保首先处理表中已删除的行，然后处理已更新的行，然后处理已插入的行。

```
 [C#]
DataTable updTable=custDS.Tables["Customers"];
// First process deletes.
custDA.Update(updTable.Select(null, null, DataViewRowState.Deleted));
// Next process updates.
custDA.Update(updTable.Select(null, null, DataViewRowState.ModifiedCurrent));
// Finally, process inserts.
custDA.Update(updTable.Select(null, null, DataViewRowState.Added));
```

7. 向 DataSet 添加现有约束

DataAdapter 的 Fill 方法仅使用数据源中的表列和表行来填充 DataSet。虽然约束通常由数据源来设置，但在默认情况下，Fill 方法不会将此架构信息添加到 DataSet 中。若要使用数据源中的现有主键约束信息填充 DataSet，则可以调用 DataAdapter 的 FillSchema 方法，或者在调用 Fill 之前将 DataAdapter 的 MissingSchemaAction 属性设置为 AddWithKey。这将确保 DataSet 中的主键约束反映数据源中的主键约束。外键约束信息不包含在内，将需要显式创建。

如果在使用数据填充 DataSet 之前向其中添加架构信息，将确保将主键约束与 DataSet 的 DataTable 对象包含在一起。这样，当再次调用 Fill 来填充 DataSet 时，将使用主键列信息来匹配数据源中的新行和每个 DataTable 中的当前行，并且使用数据源中的数据改写表中的当前数据。如果没有架构信息，来自数据源的新行将追加到 DataSet，从而导致重复的行。

注意：如果数据源中的某列被标识为自动递增列，则 MissingSchemaAction 为 AddWithKey 的 FillSchema 方法或 Fill 方法创建一个 AutoIncrement 属性设置为 true 的 DataColumn。不过，用户将需要手动设置 AutoIncrementStep 和 AutoIncrementSeed 值。

当使用 FillSchema 或将 MissingSchemaAction 设置为 AddWithKey 时，将需要在数据源中进行额外的处理来确定主键列信息。这一额外的处理可能会降低性能。如果主键信息在设计时已知，为了实现最佳性能，建议显式指定一个或多个主键列。

以下代码示例显示如何使用 FillSchema 向 DataSet 添加架构信息。

```
 [C#]
DataSet custDS=new DataSet();
custDA.FillSchema(custDS, SchemaType.Source, "Customers");
custDA.Fill(custDS, "Customers");
```

以下代码示例显示如何使用 Fill 方法的 MissingSchemaAction. AddWithKey 属性向

DataSet 添加架构信息。

```
[C#]
DataSet custDS=new DataSet();
custDA.MissingSchemaAction=MissingSchemaAction.AddWithKey;
custDA.Fill(custDS, "Customers");
```

如果 DataAdapter 遇到从 SelectCommand 中返回的多个结果集，它将在 DataSet 中创建多个表。将向这些表提供递增的默认名称 TableN，以表示 Table0 的 Table 为第一个表名。如果以参数形式向 FillSchema 方法传递表名称，则将向这些表提供递增的默认名称 TableNameN，这些表名称以表示 TableName0 的 TableName 为起始。

注意：如果对返回多个结果集的命令调用 OleDbDataAdapter 对象的 FillSchema 方法，则将只返回第一个结果集中的架构信息。当使用 OleDbDataAdapter 为多个结果集返回架构信息时，建议在调用 Fill 方法时将 MissingSchemaAction 指定为 AddWithKey 并获取架构信息。

8. 设置 DataTable 和 DataColumn 映射

DataAdapter 在其 TableMappings 属性中包含 0 个或更多个 DataTableMapping 对象的集合。DataTableMapping 提供对数据源的查询所返回的数据与 DataTable 之间的主映射。DataTableMapping 名称可以代替 DataTable 名称传递到 DataAdapter 的 Fill 方法。以下示例为 MyAuthors 表创建名为 AuthorsMapping 的 DataTableMapping。

```
[C#]
workAdapter.TableMappings.Add("AuthorsMapping", "MyAuthors");
```

DataTableMapping 使用户能够使用 DataTable 中与数据库中的列名不同的列名。当该表被更新时，DataAdapter 将使用此映射来匹配列。

如果在调用 DataAdapter 的 Fill 或 Update 方法时未指定 TableName 或 DataTableMapping 名称，DataAdapter 将查找名为 Table 的 DataTableMapping。如果 DataTableMapping 不存在，DataTable 的 TableName 将为 Table。可以通过创建名为 Table 的 DataTableMapping 来指定默认的 DataTableMapping。

以下代码示例创建一个 DataTableMapping（从 System.Data.Common 命名空间）并通过将其命名为 Table 来使其成为指定 DataAdapter 的默认映射。然后，该示例将查询结果中第一个表（Northwind 数据库的 Customers 表）中的列映射到 DataSet 的 Northwind Customers 表中的一组更为用户友好的名称。对于未映射的列，将使用数据源中的列名称。

```
[C#]
DataTableMapping custMap=custDA.TableMappings.Add("Table", "NorthwindCustomers");
custMap.ColumnMappings.Add( "CompanyName", "Company");
custMap.ColumnMappings.Add( "ContactName", "Contact");
```

```
custMap.ColumnMappings.Add( "PostalCode", "ZIPCode");
custDA.Fill(custDS);
```

在更为复杂的情况下，用户可能会决定需要使用相同的 DataAdapter 来支持为不同的表加载不同的映射。若要完成此任务，只需添加附加的 DataTableMapping 对象。

当 Fill 方法以 DataSet 实例和 DataTableMapping 名称的形式进行传递时，如果存在具有该名称的映射，则使用该映射；否则将使用具有该名称的 DataTable。

以下示例创建一个名称为 Customers，而 DataTable 名称为 BizTalkSchema 的 DataTableMapping。然后，该示例将 SELECT 语句所返回的行映射到 BizTalkSchema DataTable。

```
 [C#]
ITableMapping bizMap=custDA.TableMappings.Add("Customers", "BizTalkSchema");
bizMap.ColumnMappings.Add( "CustomerID", "ClientID");
bizMap.ColumnMappings.Add( "CompanyName", "ClientName");
bizMap.ColumnMappings.Add( "ContactName", "Contact");
bizMap.ColumnMappings.Add( "PostalCode", "ZIP");
custDA.Fill(custDS, "Customers");
```

注意：如果没有为列映射提供源列名称或者没有为表映射提供源表名称，则将自动生成默认名称。如果没有为列映射提供源列，则将给列映射提供递增的默认名称 SourceColumnN，这些名称从 SourceColumn1 开始。如果没有为表映射提供源表名称，则将给该表映射提供递增的默认名称 SourceTableN，这些名称从 SourceTable1 开始。

建议在为列映射提供源列名称时避免使用 SourceColumnN 命名约定，在为表映射提供源表名称时避免使用 SourceTableN 命名约定，因为所提供的名称可能会与 ColumnMappingCollection 中现有的默认列映射名称或 DataTableMappingCollection 中的表映射名称发生冲突。如果提供的名称已经存在，将引发异常。

如果 SelectCommand 返回多个表，Fill 将自动使用递增值为 DataSet 中的表生成表名称，这些表名称从指定表名称开始，并以 TableNameN 格式（从 TableName1 开始）继续。可以使用表映射将自动生成的表名称映射到要为 DataSet 中的表指定的名称。例如，对于返回两个表（Customers 和 Orders）的 SelectCommand，可对 Fill 发出以下调用：

```
custDA.Fill(custDS,"Customers")
```

在 DataSet 中创建了两个表：Customers 和 Customers1。可以使用表映射来确保第二个表名为 Orders 而不是 Customers1。若要完成此任务，则将 Customers1 的源表映射到 DataSet 表 Orders，如以下示例所示：

```
custDA.TableMappings.Add("Customers1", "Orders")
custDA.Fill(custDS, "Customers")
```

9. 将参数用于 DataAdapter

DataAdapter 具有 4 项用于从数据源检索数据和向数据源更新数据的属性。SelectCommand 属性从数据源中返回数据。InsertCommand、UpdateCommand 和 DeleteCommand 属性用于管理数据源中的更改。在调用 DataAdapter 的 Fill 方法之前，必须设置 SelectCommand 属性。根据对 DataSet 中的数据作出的更改，在调用 DataAdapter 的 Update 方法之前，必须设置 InsertCommand、UpdateCommand 或 DeleteCommand 属性。例如，如果已添加行，在调用 Update 之前必须设置 InsertCommand。当 Update 处理已插入、更新或删除的行时，DataAdapter 将使用相应的 Command 属性来处理该操作。有关已修改行的当前信息将通过 Parameters 集合传递到 Command 对象。

例如，当更新数据源中的行时，将调用 UPDATE 语句，它使用唯一标识符来表示该表中要更新的行。该唯一标识符通常是主键字段的值。UPDATE 语句使用既包含唯一标识符又包含要更新的列和值的参数，如以下 SQL 语句所示：

```
UPDATE Customers SET CompanyName=@CompanyName WHERE CustomerID=@CustomerID
```

在该示例中，将其中 CustomerID 等于@CustomerID 参数值的行的 CompanyName 字段用@CompanyName 参数值来进行更新。这些参数使用 Parameter 对象的 SourceColumn 属性从已修改的行中检索相关信息。下面是上一示例 UPDATE 语句的参数：

```
custDA.Parameters.Add("@CompanyName", SqlDbType.NChar, 15, "CompanyName")
Dim myParm As SqlParameter=custDA.UpdateCommand.Parameters.Add("@CustomerID", _
                                        SqlDbType.NChar, 5, "CustomerID")
myParm.SourceVersion=DataRowVersion.Original
```

Parameters 集合的 Add 方法采用参数的名称、DataAdapter 特定类型、大小（如果可应用于该类型）及 DataTable 中 SourceColumn 的名称。注意，@CustomerID 参数的 SourceVersion 设置为 Original。这样，如果标识列的值已经在修改后的 DataRow 中被更改，就一定会更新数据源中的现有行。在这种情况下，Original 行值将匹配数据源中的当前值，而 Current 行值将包含更新的值。如果没有为@CompanyName 参数设置 SourceVersion，而将使用默认的 Current 行值。

以下示例显示要用作 DataAdapter 的 SelectCommand、InsertCommand、UpdateCommand 和 DeleteCommand 属性的 CommandText 的示例 SQL 语句。对于 OleDbDataAdapter 对象和 OdbcDataAdapter 对象，必须使用占位符来标识参数。对于 SqlDataAdapter 对象，必须使用命名参数。

```
[C#]
string selectSQL="SELECT CustomerID, CompanyName FROM Customers WHERE Country=@Country
AND City=@City";
```

```
string insertSQL="INSERT INTO Customers (CustomerID, CompanyName) "+
                "VALUES (@CustomerID, @CompanyName)";
string updateSQL="UPDATE Customers SET CustomerID=@CustomerID, CompanyName=
@CompanyName "+"WHERE CustomerID=@OldCustomerID";
string deleteSQL="DELETE FROM Customers WHERE CustomerID=@CustomerID";
```

参数化查询语句定义将需要创建哪些输入和输出参数。若要创建参数，则使用 Parameters. Add 方法或 Parameter 构造函数来指定列名称、数据类型和大小。对于内部数据类型(如 Integer)，无须包含大小或者可以指定默认大小。

以下代码示例为上一示例中的 SQL 语句创建参数并填充 DataSet。

```
[C#]
SqlConnection nwindConn=new SqlConnection("Data Source=localhost;Integrated Security=
SSPI;Initial Catalog=northwind");
SqlDataAdapter custDA=new SqlDataAdapter();
SqlCommand selectCMD=new SqlCommand(selectSQL, nwindConn);
custDA.SelectCommand=selectCMD;
// Add parameters and set values.
selectCMD.Parameters.Add("@Country", SqlDbType.NVarChar, 15).Value="UK";
selectCMD.Parameters.Add("@City", SqlDbType.NVarChar, 15).Value="London";
DataSet custDS=new DataSet();
custDA.Fill(custDS, "Customers");
```

10. Parameter. DbType

参数的类型是特定于. NET Framework 数据提供程序的。如果指定类型，则在向数据源传递 Parameter 的值之前，会将该值转换为. NET Framework 数据提供程序类型。如果未指定类型，ADO. NET 将从 Parameter 对象的 Value 的. NET Framework 类型推断 Parameter 的. NET Framework 数据提供程序类型。也可以通过将 Parameter 对象的 DbType 属性指定为特定的 System. Data. DbType，以一般的方式来指定 Parameter 的类型。此外，ADO. NET 将从 Parameter 对象的 DbType 推断 Parameter 的. NET Framework 数据提供程序类型。Parameter 对象的. NET Framework 数据提供程序类型将从 Parameter 对象的 Value 的. NET Framework 类型来推断，或从 Parameter 对象的 DbType 来推断。表 4-4 显示了根据作为 Parameter 值或指定的 DbType 传递的对象推断出的 Parameter 类型。

注意：随. NET Framework 1. 0 版提供的. NET Framework 数据提供程序不验证 Decimal 参数值的 Precision 和 Scale，这可能会导致截断的数据被插入数据源。如果使用的是. NET Framework 1. 0 版，请在设置参数值之前，验证 Decimal 值的 Precision 和 Scale。

表 4-4　Parameter 类型

类　　型	System. Data. DbType	SqlDbType
bool	Boolean	Bit
byte	Byte	TinyInt
byte[]	Binary	VarBinary。如果字节数组大于 8000 字节，此隐式转换将失败。对于大于 8000 字节的字节数组，则显式设置 SqlDbType
char		不支持从 char 推断 SqlDbType
DateTime	DateTime	DateTime
Decimal	Decimal	Decimal
double	Double	Float
float	Single	Real
Guid	Guid	UniqueIdentifier
Int16	Int16	SmallInt
Int32	Int32	Int
Int64	Int64	BitInt
object	Object	Variant
string	String	NVarChar。如果字符串大于 4000 字节，此隐式转换将失败。对于大于 4000 字符的字符串，请显式设置 SqlDbType
TimeSpan	Time	不支持从 TimeSpan 推断 SqlDbType
UInt16	UInt16	不支持从 UInt16 推断 SqlDbType
UInt32	UInt32	不支持从 UInt32 推断 SqlDbType
UInt64	UInt64	不支持从 UInt64 推断 SqlDbType

在.NET Framework 1.1 版及更高版本中，当用无效的 Precision 设置 Decimal 参数值时会发生异常。但仍将截断超出 Decimal 参数小数位数的 Scale 值。

11. Parameter. Direction

表 4-5 显示可以与 ParameterDirection 枚举一起用来设置 Parameter 的 Direction 的值。

表 4-5　Direction 值

成员名称	说　　明
Input	该参数是输入参数。这是默认设置
InputOutput	该参数能够输入和输出
Output	该参数是输出参数
ReturnValue	该参数表示返回值

以下代码示例显示如何设置 Parameter 的 Direction。

```
myParm.Direction=ParameterDirection.Output
```

SourceColumn 和 SourceVersion 可以作为参数传递到 Parameter 构造函数，或设置为现有 Parameter 的属性。SourceColumn 是来自将在其中检索 Parameter 值的 DataRow 的 DataColumn 的名称。SourceVersion 指定 DataAdapter 使用哪个 DataRow 版本来检索值。

表 4-6 显示可以与 SourceVersion 一起使用的 DataRowVersion 枚举值。

表 4-6 DataRowVersion 枚举值

成员名称	说明
Current	该参数使用列的当前值。这是默认设置
Default	该参数使用列的 DefaultValue
Original	该参数使用列的初始值
Proposed	该参数使用建议值

以下代码示例定义一个 UPDATE 语句，在该语句中，CustomerID 列用作以下两个参数的 SourceColumn：@CustomerID(SET CustomerID=@CustomerID)和@OldCustomerID(WHERE CustomerID=@OldCustomerID)。@CustomerID 参数用于将 CustomerID 列更新为 DataRow 中的当前值。因此，将使用 SourceVersion 为 Current 的 CustomerID SourceColumn。@OldCustomerID 参数用于标识数据源中的当前行。由于在该行的 Original 版本中找到了匹配列值，所以将使用 SourceVersion 为 Original 的相同 SourceColumn(CustomerID)。

```
[C#]
custDA.UpdateCommand.Parameters.Add("@CustomerID", SqlDbType.NChar, 5, "CustomerID");
custDA.UpdateCommand.Parameters.Add("@CompanyName", SqlDbType.NVarChar, 40, "Com-
panyName");
SqlParameter myParm=custDA.UpdateCommand.Parameters.Add("@OldCustomerID", SqlDbType.
NChar, 5, "CustomerID");
myParm.SourceVersion=DataRowVersion.Original;
```

12. UpdateDRowSource

可以使用 Command 对象的 UpdatedRowSource 属性控制从数据源中返回的值如何映射回 DataSet。通过将 UpdateDRowSource 属性设置为 UpdateRowSource 枚举值之一，可以控制是忽略 DataAdapter 命令返回的参数还是将其应用于 DataSet 中已更改的行。还可以指定是否将返回的第一个行(如果存在)应用于 DataSet 中已更改的行。

表 4-7 描述了 UpdateRowSource 枚举的不同值，并说明它们如何影响与 DataAdapter 一起使用的命令的行为。

表 4-7　UpdateRowSource 的枚举值

UpdateRowSource	说　明
Both	输出参数和返回结果集的第一行都可以映射到 DataSet 中已更改的行
FirstReturnedRecord	只有返回的结果集第一行中的数据才可以映射到 DataSet 中已更改的行
None	将忽略任何输出参数或返回结果集的行
OutputParameters	只有输出参数才可以映射到 DataSet 中已更改的行

13. 输入和输出参数及返回值

除了输入和输出参数之外，存储过程还可以具有返回值。以下示例阐释 ADO.NET 如何发送和接收输入参数、输出参数和返回值，其中采用了这样一种常见方案：将新记录插入其中主键列是自动编号字段的表。该示例使用输出参数来返回自动编号字段的@@Identity，而 DataAdapter 则将其绑定到 DataTable 的列，使 DataSet 反映所生成的主键值。

该示例使用以下存储过程将新目录插入 Northwind Categories 表（该表将 CategoryName 列中的值当作输入参数），从@@Identity 中以输出参数的形式返回自动编号字段 CategoryID 的值，并提供所影响行数的返回值。

```
CREATE PROCEDURE InsertCategory
  @CategoryName nchar(15),
  @Identity int OUT
AS
INSERT INTO Categories (CategoryName) VALUES(@CategoryName)
SET @Identity=@@Identity
RETURN @@ROWCOUNT
```

以下示例将 InsertCategory 存储过程用作 DataAdapter 的 InsertCommand 的数据源。通过将 CategoryID 列指定为@Identity 输出参数的 SourceColumn，当调用 DataAdapter 的 Update 方法时，所生成的自动编号值将在该记录插入数据库后在 DataSet 中得到反映。

对于 OleDbDataAdapter，必须在指定其他参数之前先指定 ParameterDirection 为 ReturnValue 的参数。

```
[C#]
SqlConnection nwindConn=new SqlConnection("Data Source=localhost;Integrated Security=
SSPI;"+"Initial Catalog=northwind");
SqlDataAdapter catDA=new SqlDataAdapter("SELECT CategoryID, CategoryName FROM Catego-
ries", nwindConn);
catDA.InsertCommand=new SqlCommand("InsertCategory", nwindConn);
```

```
catDA.InsertCommand.CommandType=CommandType.StoredProcedure;
SqlParameter myParm=catDA.InsertCommand.Parameters.Add("@RowCount", SqlDbType.Int);
myParm.Direction=ParameterDirection.ReturnValue;
catDA.InsertCommand.Parameters.Add("@CategoryName", SqlDbType.NChar, 15, "Catego-
ryName");
myParm=catDA.InsertCommand.Parameters.Add("@Identity", SqlDbType.Int, 0, "Category-
ID");
myParm.Direction=ParameterDirection.Output;
DataSet catDS=new DataSet();
catDA.Fill(catDS, "Categories");
DataRow newRow=catDS.Tables["Categories"].NewRow();
newRow["CategoryName"]="New Category";
catDS.Tables["Categories"].Rows.Add(newRow);
catDA.Update(catDS, "Categories");
Int32 rowCount=(Int32)catDA.InsertCommand.Parameters["@RowCount"].Value;
```

4.3.4 DataReader 对象

1. 使用 DataReader 检索数据

可以使用 ADO.NET DataReader 从数据库中检索只读、只进的数据流。查询结果在查询执行时返回,并存储在客户端的网络缓冲区中,直到用户使用 DataReader 的 Read 方法对它们发出请求。使用 DataReader 可以提高应用程序的性能,因为一旦数据可用,DataReader 方法就立即检索该数据,而不是等待返回查询的全部结果。并且在默认情况下,该方法一次只在内存中存储一行,从而降低了系统开销。

当创建 Command 对象的实例后,可调用 Command.ExecuteReader 从数据源中检索行,从而创建一个 DataReader,如以下示例所示:

```
[C#]
SqlDataReader myReader=myCommand.ExecuteReader();
```

使用 DataReader 对象的 Read 方法可从查询结果中获取行。通过向 DataReader 传递列的名称或序号引用,可以访问返回行的每一列。不过,为了实现最佳性能,DataReader 提供了一系列方法,它们将使用户能够访问其本机数据类型(GetDateTime、GetDouble、GetGuid 和 GetInt32 等)形式的列值。在基础数据类型为已知时,如果使用类型化访问器方法,将减少在检索列值时所需的类型转换量。

注意: *.NET Framework 1.1 版包含 DataReader 的附加属性 HasRows,该属性使用户在读取 DataReader 之前就可确定它是否返回了任何结果。*

以下代码示例循环访问一个 DataReader 对象,并从每个行中返回两个列。

```
[C#]
if (myReader.HasRows)
  while (myReader.Read())
    Console.WriteLine("\t{0}\t{1}", myReader.GetInt32(0), myReader.GetString(1));
else
  Console.WriteLine("No rows returned.");
myReader.Close();
```

DataReader 提供未缓冲的数据流，该数据流使过程逻辑可以有效地按顺序处理从数据源中返回的结果。由于数据不在内存中缓存，所以在检索大量数据时，DataReader 是一种适合的选择。

2. 关闭 DataReader

每次使用完 DataReader 对象后都应调用 Close 方法。

如果 Command 包含输出参数或返回值，那么在 DataReader 关闭之前，将无法访问这些输出参数或返回值。

注意：当 DataReader 打开时，该 DataReader 将以独占方式使用 Connection。在初始 DataReader 关闭之前，将无法对 Connection 执行任何命令（包括创建另一个 DataReader）。

不要在类的 Finalize 方法中对 Connection、DataReader 或任何其他托管对象调用 Close 或 Dispose。在终结器中，仅释放类直接拥有的非托管资源。如果类不拥有任何非托管资源，则不要在类定义中包含 Finalize 方法。

3. 多个结果集

如果返回的是多个结果集，DataReader 会提供 NextResult 方法来按顺序循环访问这些结果集，如以下代码示例所示：

```
[C#]
SqlCommand myCMD=new SqlCommand("SELECT CategoryID, CategoryName FROM Categories;"+
                                  "SELECT EmployeeID, LastName FROM Employees",
nwindConn);
nwindConn.Open();
SqlDataReader myReader=myCMD.ExecuteReader();
do
{
  Console.WriteLine("\t{0}\t{1}", myReader.GetName(0), myReader.GetName(1));
  while (myReader.Read())
    Console.WriteLine("\t{0}\t{1}", myReader.GetInt32(0), myReader.GetString(1));
} while (myReader.NextResult());
myReader.Close();
nwindConn.Close();
```

4. 从 DataReader 中获取架构信息

当 DataReader 打开时，可以使用 GetSchemaTable 方法检索有关当前结果集的架构信息。GetSchemaTable 将返回一个填充了行和列的 DataTable 对象，这些行和列包含当前结果集的架构信息。对于结果集的每一列，DataTable 都将包含一行。架构表行的每一列都映射到在结果集中返回的列的属性，其中 ColumnName 是属性的名称，而列的值为属性的值。以下代码示例为 DataReader 写出架构信息。

```
 [C#]
DataTable schemaTable=myReader.GetSchemaTable();
foreach (DataRow myRow in schemaTable.Rows)
{
  foreach (DataColumn myCol in schemaTable.Columns)
    Console.WriteLine(myCol.ColumnName+"="+myRow[myCol]);
  Console.WriteLine();
}
```

5. SqlDataReader 类

提供一种从数据库读取只进的行流的一种方式。无法继承此类。

以下示例创建一个 SqlConnection、一个 SqlConnection 和一个 SqlDataReader。该示例读取全部数据，并将这些数据写到控制台。最后，该示例先关闭 SqlDataReader，然后关闭 SqlConnection。

```
[C#]
public void ReadMyData(string myConnString) {
    string mySelectQuery="SELECT OrderID, CustomerID FROM Orders";
    SqlConnection myConnection=new SqlConnection(myConnString);
    SqlCommand myCommand=new SqlCommand(mySelectQuery,myConnection);
    myConnection.Open();
    SqlDataReader myReader;
    myReader=myCommand.ExecuteReader();
    // Always call Read before accessing data.
    while (myReader.Read()) {
      Console.WriteLine(myReader.GetInt32(0)+", "+myReader.GetString(1));
    }
    // always call Close when done reading.
    myReader.Close();
    // Close the connection when done with it.
    myConnection.Close();
}
```

通过本章的学习，读者应该能够通过 ADO.NET 访问各种数据库，能够对数据库进行增、删、查和改等操作，能够在 ADO.NET 中调用存储过程。

第 5 章　远程教育系统实例

5.1　系统规划与设计

前面几章讲述了网站建设的基本知识。作为这些知识的综合应用，本章将介绍一个具体的实例——远程教育系统。现代远程教育的发展已经成为全球性的大趋势，远程开放教育成为各级各类学校的一种重要教育手段，实施远程开放教育的大学继续在扩大规模、提高质量上发挥作用，并将成为终身教育和学习化社会的重要组成部分。这里所涉及的远程教育系统的实例，具体的介绍如下。

5.1.1　系统平台

在进行系统设计之前，首先明确一下系统开发平台和应用平台。

开发平台如下。

(1) Microsoft Visual Studio.Net2005，使用 C#语言。

(2) Microsoft IIS 6.0。

(3) Microsoft SQL Server 2000。

应用平台如下。

(1) Microsoft .NET Framework。

(2) Microsoft IIS 6.0。

(3) Microsoft SQL Server 2000。

5.1.2　系统功能概述

我们要实现的是一个具有基本功能的远程教育系统。那么对于一个典型的远程教育系统来说，都有哪些基本的功能呢？

首先进入网站首页，上面有多个栏目，文章按栏目分类。前台提供的主要功能如下。

(1) 以各种角色(管理员、教师和学员)登录进入管理平台的登录窗口。

(2) 学习简报栏目。

(3) 答疑解惑栏目。

(4) 友情链接栏目。

（5）系统简介栏目。

（6）培训公告栏目。

（7）作业展示栏目。

（8）思考习题栏目。

（9）导航栏中的系统主页、系统简介、培训公告、学习简报、答疑解惑、作业展示、思考习题和友情链接等二级页面的链接。

对于各个栏目，由栏目首页、含栏目相关文章标题列表等页面组成，单击标题可以查看到文章的相关内容。文章详细内容显示页面将显示相应文章的标题、作者、发表时间、文章正文等内容。

后台提供的功能，如表 5-1 所示。

表 5-1　角色及对应的功能

角　色	功　　能
管理员	系统简介管理，培训公告管理，学习简报管理，答疑解惑管理，课程管理，友情链接管理，用户管理，个人信息设置，退出
教师	作业展示管理，思考习题管理，个人信息设置，退出
学员	作业展示管理，思考习题管理，个人信息设置，退出

以上是对系统进行简单的介绍，在接下来的章节中将进行详细讲述。

5.1.3　系统模块划分与流程

根据上面的功能列表，可归纳出系统的模块划分，如表 5-2 所示。

表 5-2　模块功能列表

序号	功能模块类别	功 能 模 块	备　　注
1	系统类（对常用功能的封装）	基础数据连接类	提供对数据库连接的支持
2		基础数据操作类	提供对数据操作的支持
3		业务数据操作类	提供对具体业务数据操作的支持
4	浏览者（前台用户）	首页模块	
5		栏目页模块	
6		文章浏览模块	
7		登录模块	
8	管理员（后台用户）	系统简介管理模块	
9		培训公告管理模块	
10		学习简报管理模块	
11		答疑解惑管理模块	

续表

序号	功能模块类别	功 能 模 块	备　注
12	管理员（后台用户）	课程管理模块	
13		友情链接管理模块	
14		用户管理模块	
15		个人信息设置模块	
16	教师（后台用户）	作业展示管理模块	
17		思考习题管理模块	
18		个人信息设置模块	
19	学员（后台用户）	作业展示管理模块	
20		思考习题管理模块	
21		个人信息设置模块	

可以看到，需求一步一步地明确了。下面为了更清晰地表现出需求的细节，将根据功能概述和模块划分，画出系统流程图。首先画出大功能类别的流程图。

图 5-1 所示为前台用户模块流程图，图 5-2 所示为管理员模块流程图，图 5-3 所示为教师模块流程图，图 5-4 所示为学员模块流程图。

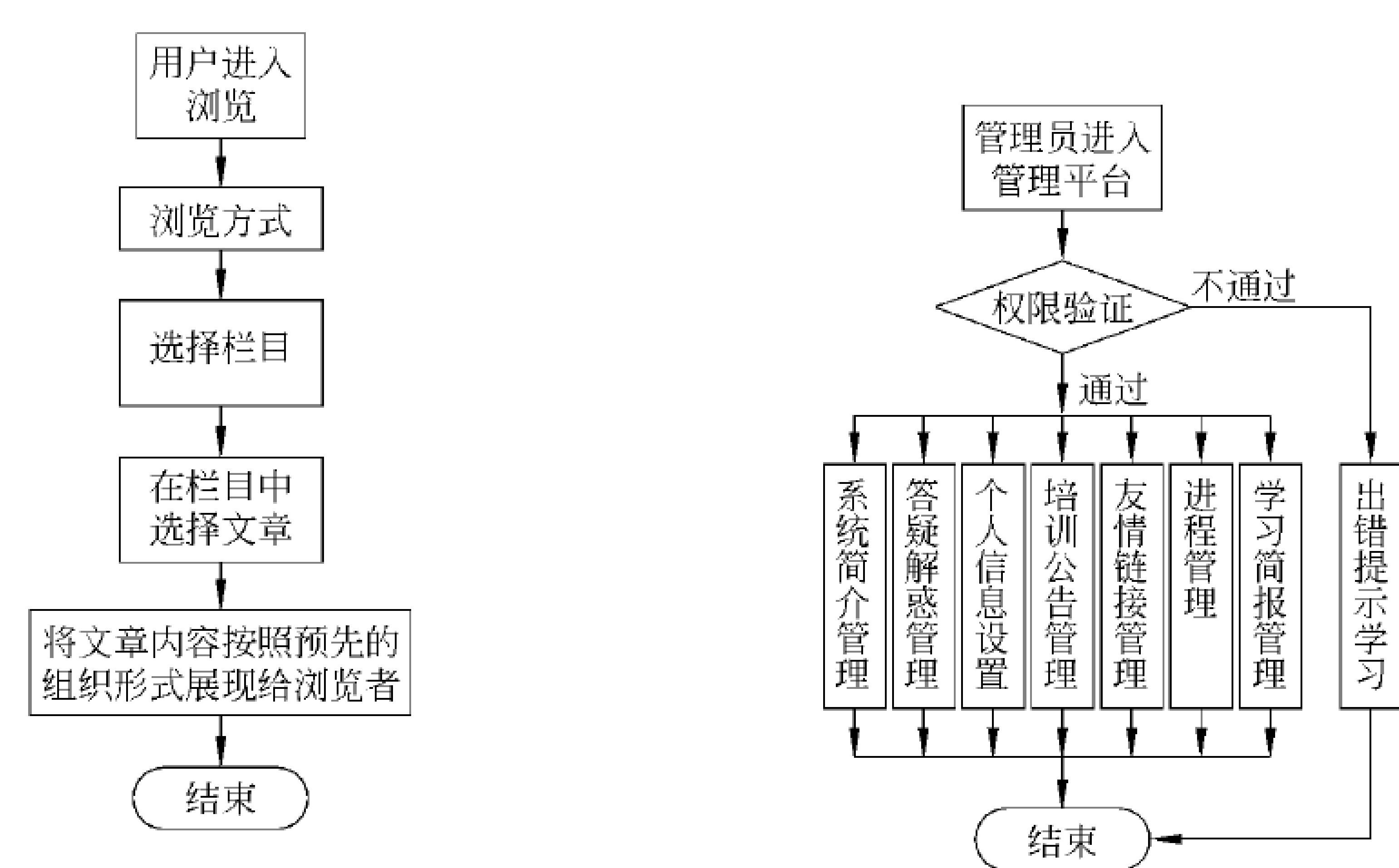


图 5-1　前台用户模块流程图

图 5-2　管理员模块流程图

以上就是系统 4 大模块的整体流程。下一节将进一步对功能进行细化。

5.1.4　系统功能设计与列表

此节将进入到详细设计阶段。根据 5.1.3 节中流程图和模块的分析及划分结果，在这一节将把每个模块的详细功能以表的形式列出来，将各个模块之间的结构关系整理清

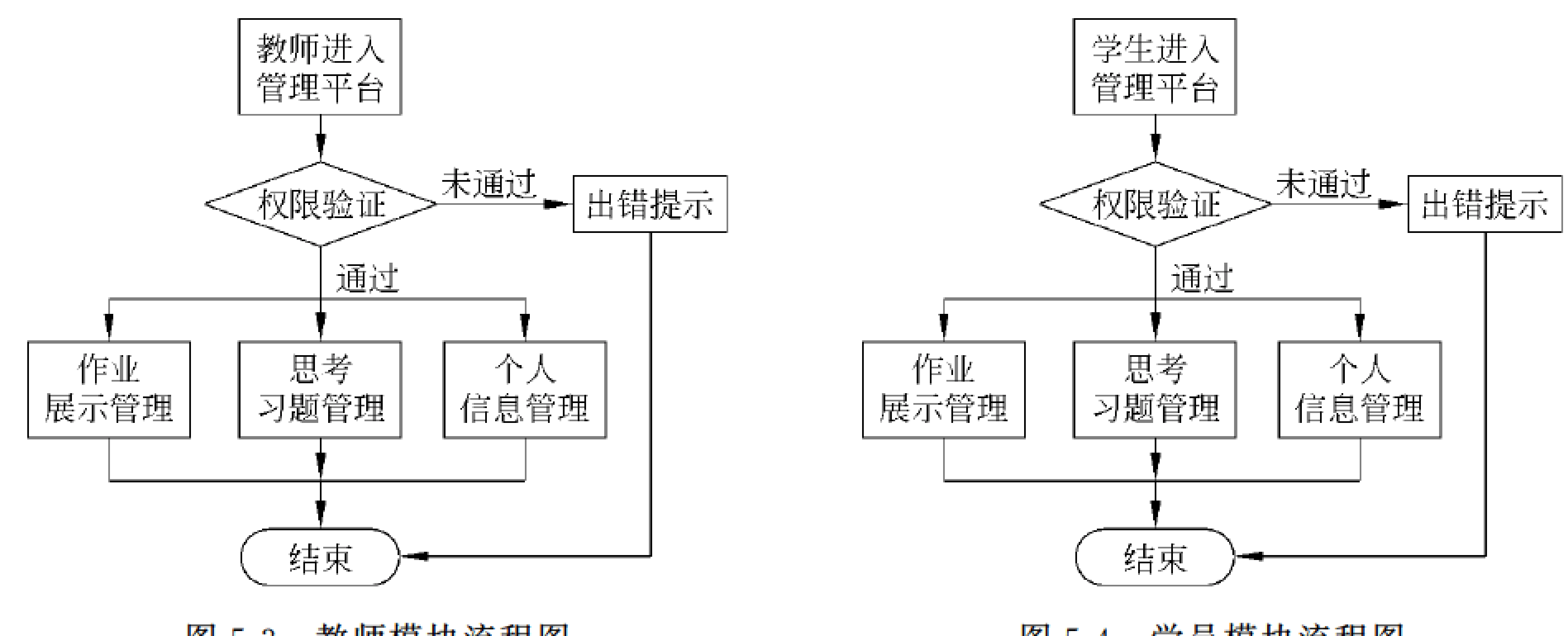


图 5-3　教师模块流程图　　　　图 5-4　学员模块流程图

楚，这将为后面章节中的编码阶段打下良好的基础。表 5-2 所示共有 18 个模块，前台 4 个模块，后台 14 个模块。下面开始详细描述模块功能和流程（如表 5-3～表 5-20 所示）。

表 5-3　首页模块功能

序号	功能列表	功 能 明 细	备注
1	学习简报	按发布学习简报的时间排序取最新的 4 篇学习简报	
2	答疑解惑	按发布答疑解惑的时间排序取最新的 4 篇答疑解惑	
3	友情链接	按建立友情链接的时间排序取最新的 4 个友情链接	
4	系统简介	对系统的简单介绍，按规定显示字的个数	
5	培训公告	按发布培训公告的时间排序取最新的 5 篇培训公告	
6	作业展示	按学生作业的分数排序取分数最高的 5 篇作业答卷	
7	思考习题	按学生思考习题的分数排序取分数最高的 5 篇思考习题答卷	

首页模块实际上只有 1 个页面，上面进行的是栏目的划分。

表 5-4　栏目页模块功能

序号	功能列表	功 能 明 细	备注
1	学习简报列表	单击“更多”时进入学习简报列表，按照时间排序列出了所有的学习简报	
2	答疑解惑列表	单击“更多”时进入答疑解惑列表，按照时间排序列出了所有的答疑解惑	
3	友情链接列表	单击“更多”时进入友情链接列表，按照时间排序列出了所有的友情链接	
4	系统简介主页	单击“查看”时进入系统简介主页，显示了系统简介的完整内容	
5	培训公告列表	单击“更多”时进入培训公告列表，按照时间排序列出了所有的培训公告	
6	作业展示列表	单击“更多”时进入作业展示列表，按照分数排序列出了所有的作业展示	
7	思考习题列表	单击“更多”时进入思考习题列表，按照分数排序列出了所有的思考习题	

栏目页模块有 7 个页面，每个栏目 1 个页面。

表 5-5　文章浏览模块功能

序号	功能列表	功能明细	备注
1	学习简报内容显示	文章的详细内容显示，包括标题、发布者、发布时间和正文等	
2	答疑解惑内容显示	文章的详细内容显示，包括标题、发布者、发布时间和正文等	
3	培训公告内容显示	文章的详细内容显示，包括标题、发布者、发布时间和正文等	
4	作业展示内容显示	文章的详细内容显示，包括作业标题、学号、姓名、作业要求、作业内容、教师评语和得分等	
5	思考习题内容显示	文章的详细内容显示，包括习题标题、学号、姓名、习题要求、习题内容、教师评语和得分等	

文章页模块实际上有 5 个页面，不同栏目的文章用不同的页面显示。

表 5-6　登录模块功能

序号	功能列表	功能明细	备注
1	登录	通过选择角色、输入用户名和密码登录到管理平台	
2	注册	新用户注册个人信息（只限于学员注册）	
3	找回密码	用户可以通过输入真实姓名和身份证号码找回密码	

表 5-7　系统简介管理模块（管理员）功能

序号	功能列表	功能明细	备注
1	发布系统简介	如果数据库中没有系统简介的内容，管理员可以发布新的系统简介的内容	
2	修改系统简介	管理员可以对原有的系统简介的内容进行修改	
3	删除系统简介	管理员可以删除原有的系统简介的内容	

表 5-8　培训公告管理模块（管理员）功能

序号	功能列表	功能明细	备注
1	新增培训公告	管理员可以增加新的培训公告	
2	修改培训公告	管理员可以修改原有的培训公告	
3	删除培训公告	管理员可以删除原有的培训公告	

表 5-9　学习简报管理模块(管理员)功能

序号	功能列表	功 能 明 细	备注
1	新增学习简报	管理员可以增加新的学习简报	
2	修改学习简报	管理员可以修改原有的学习简报	
3	删除学习简报	管理员可以删除原有的学习简报	

表 5-10　答疑解惑管理模块(管理员)功能

序号	功能列表	功 能 明 细	备注
1	新增答疑解惑	管理员可以增加新的答疑解惑	
2	修改答疑解惑	管理员可以修改原有的答疑解惑	
3	删除答疑解惑	管理员可以删除原有的答疑解惑	

表 5-11　课程管理模块(管理员)功能

序号	功能列表	功 能 明 细	备注
1	查询	管理员通过输入课程的名称可以查询到该课程的任课教师。如果不输入课程名称,默认的是查询所有课程的名称及任课教师	
2	新增课程	管理员可以新增加课程	
3	修改	管理员可以修改课程名称及任课教师	
4	删除	管理员可以删除课程名称和任课教师	

表 5-12　友情链接管理模块(管理员)功能

序号	功能列表	功 能 明 细	备注
1	新增链接	管理员可以增加新的链接	
2	修改链接	管理员可以修改原有的链接	
3	删除链接	管理员可以删除原有的链接	

表 5-13　用户管理模块(管理员)功能

序号	功能列表	功 能 明 细	备注
1	查询	管理员可以通过输入用户名查询到该用户的基本情况(密码和角色名称),如果不输入用户名,默认的是查询所有用户的基本情况	
2	新增用户	管理员可以增加新的用户(管理员和教师)	
3	删除	管理员可以删除用户	

表 5-14　个人信息设置(管理员)功能

序号	功能列表	功 能 明 细	备注
1	设置个人信息	设置管理员自己的个人信息	

表 5-15　作业展示管理(教师)功能

序号	功能列表	功 能 明 细	备注
1	查询作业	教师选择课程以后,可查询到自己发布的关于该课程的作业,可以修改和删除该作业,单击作业的标题可以查看到学生做该作业的情况	
2	新增作业	教师可以增加新的作业	

表 5-16　思考习题管理(教师)功能

序号	功能列表	功 能 明 细	备注
1	查询思考习题	教师选择课程以后,可查询到自己发布的关于该课程的习题,可以修改和删除该习题,单击习题的标题可以查看到学生做该习题的情况	
2	新增思考习题	教师可以增加新的习题	

表 5-17　个人信息设置(教师)功能

序号	功能列表	功 能 明 细	备注
1	设置个人信息	设置教师自己的个人信息	

表 5-18　作业展示管理(学员)功能

序号	功能列表	功 能 明 细	备注
1	查询作业	学员可以按照教师或者课程两个查询条件进行查询,单击作业的名称可以查看到该作业,并且可以回答作业	

表 5-19　思考习题管理(学员)功能

序号	功能列表	功 能 明 细	备注
1	查询习题	学员可以按照教师或者课程两个查询条件进行查询,单击习题的名称可以查看到该习题,并且可以回答习题	

表 5-20　个人信息设置(学员)功能

序号	功能列表	功 能 明 细	备注
1	设置个人信息	设置学员自己的个人信息	

5.1.5　系统功能的扩充

系统功能的扩充是系统开发中非常重要的一个环节,因为在当前现实的项目开发中,

几乎没有一个项目开发一次就满足所有需求，后期的系统升级是必需的。所以建立系统的架构和设计系统的时候，一定要注意系统的可扩充性，而且现在很多的项目开发是分期进行的。本书实例中的一些列表功能是给读者作为参考来使用的。

下一节的数据设计，可以说是系统架构和扩充的重要基础，它决定系统的许多关键部件。

5.2 数据库设计

数据库设计是项目开发中系统设计的另一个关键环节，在这里之所以要特别强调数据库设计的重要性，是因为数据库设计就像高楼大厦的根基一样，如果设计不好，在后来的系统维护、变更和功能扩充时，甚至于在系统开发过程中，将会引起比较大的问题，大量的工作将会重新进行。

下面根据 5.1 节的分析，开始设计相关的数据库。

5.2.1 数据库表及表之间的相互关系

基于上述的详细设计，需要设计到下列各表，这些表之间相互关联，共同存储着系统所需的数据。在设计数据库表的过程中，可以记住几条原则，数据库设计一个表最好只存储一个实体或对象的相关信息，不同的实体最好存储在不同的数据表中，如果实体还可以再划分，实体的划分原则是最好能够比当前系统开发的实体颗粒要小；数据表的信息结构一定要合适，表的字段数量一定不要过多；扩充信息和动态变化的信息一定要分开在不同的表中；对于出现多对多这样的表关系系统，尽量不出现等。在本系统中所需要设计的数据库表如表 5-21 所示。

表 5-21 数据库表

序号	数 据 库 表	数据库表存储内容
1	Role	存储用户所属的角色信息
2	User	存储用户基本信息
3	Column	存储系统栏目信息
4	Article	存储文章信息
5	Course	存储课程信息
6	HomeWork	存储作业和思考习题信息
7	Answer	存储学员的答题信息
8	FriendshipLink	存储友情链接的信息

表 5-21 列出了根据原则设计出的系统所要使用到的数据库表，并根据流程和功能以图的形式表示出数据库各表之间的相互关系，具体如图 5-5 所示。

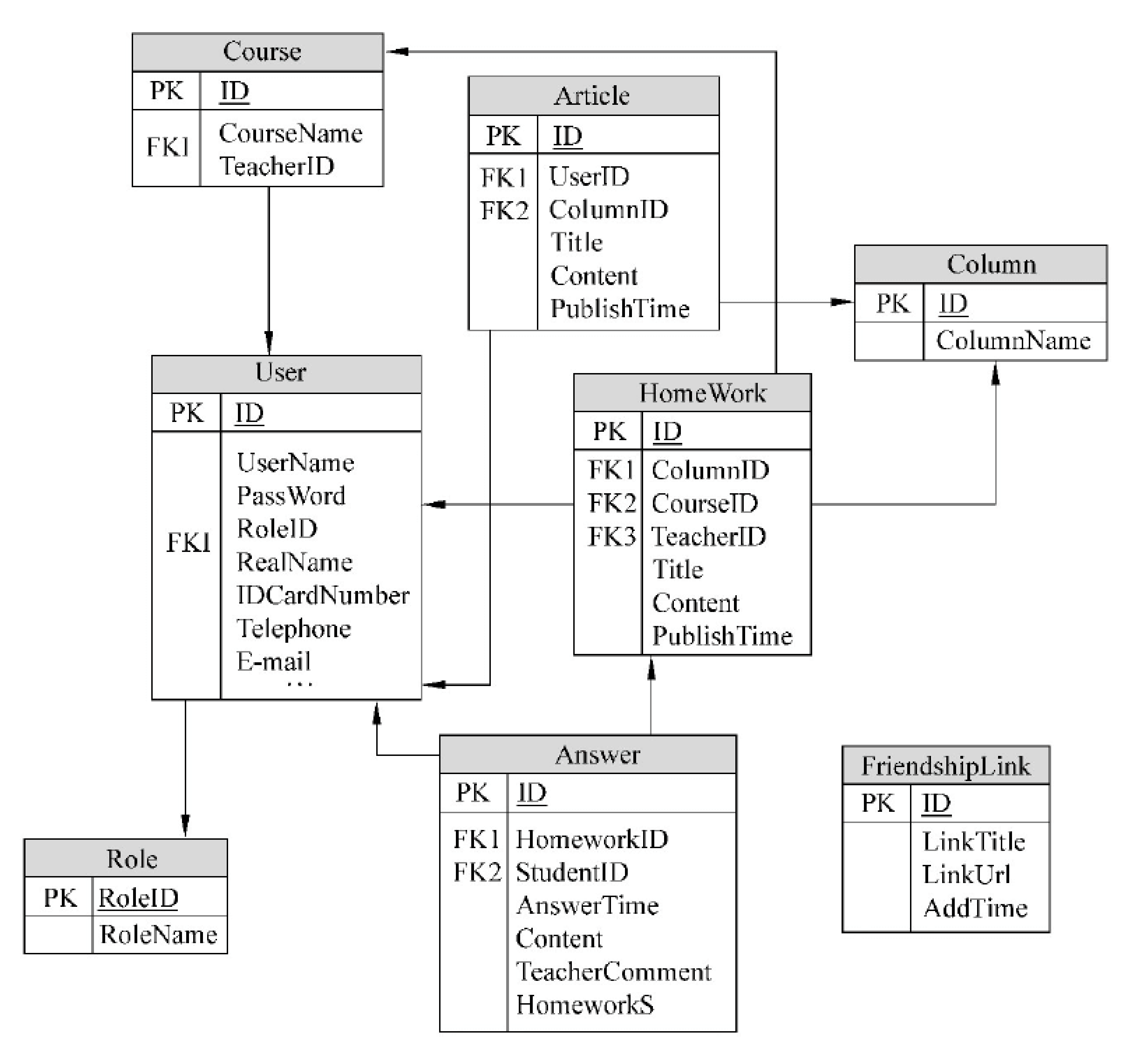


图 5-5　数据表之间的相互关系

5.2.2　数据库结构的详细设计

根据数据量的大小不同，系统可以使用不同的数据库。本系统使用的是 SQL Server 2000 数据库。SQL Server 数据库使用方便，数据存储量也比较大，是比较适合中小型网站开发的数据库。对于大数据量的网站，可以使用 Oracle 数据库。

远程教育系统的数据库中各表的设计结果如下面几个表格所示。每个表格表示在数据库中的一个表。接下来将根据设计的逻辑关系和目的逐一介绍各数据表的设计思路。

1. 角色表(Role)设计

此数据表是系统的基础表，系统投入开始使用之前，就已经进行了数据的初始化，此表的设计便于系统的扩展。

表 5-22 所示为用户角色表，本数据表的设计是为了满足系统以后功能的扩展需求，如果系统需要进行扩展，增加其他的用户角色，则只需在此表中添加相应的角色名称，配置该角色对应的权限(本系统是通过 XML 配置角色的权限，稍后章节会讲到)，再在用户表中增加对应角色的用户，就能够实现系统的完美扩展。

表 5-22 用户角色表

序号	字段含义	字段名称	类型	备　注
1	角色编号	RoleID	int	主键,需设置为自增长
2	角色名称	RoleName	varchar	

表 5-22 中的各字段说明如下。

- 角色编号：是角色表的主键,唯一表示一个角色。
- 角色名称：是用户所属角色的名称,如学员、教师和系统管理员等。

表 5-22 在初始化时,初始了如表 5-23 所示的数据,为方便系统投入运行后,标识用户所属的角色。

表 5-23 用户角色表初始化数据

角色编号(RoleID)	角色名称(RoleName)
1	系统管理员
2	教师
3	学员

2. 用户表(User)设计

表 5-24 所示为 User 用户表,用于存放系统的用户信息。由于用户表中字段较多,且大多数是用户的个人信息,所以表 5-24 只是列出了常用的字段及与其他表有关联的字段。

表 5-24 用户表

序号	字段含义	字段名称	类型	备　注
1	用户编号	ID	int	主键,需设置为自增长
2	用户名	UserName	varchar	作为用户登录用
3	密码	PassWord	varchar	
4	角色编号	RoleID	int	与角色表关联
5	真实姓名	RealName	varchar	
6	身份证号	IDCardNumber	varchar	
7	联系电话	Telephone	varchar	
8	电子邮件	E-mail	varchar	

表 5-24 中的各字段说明如下。

- 用户编号(ID)：是用户表的主键,唯一标识一个用户,在其他表中都需要关联它,例如课程表(course)就要使用教师编号(TeacherID)来建立课程与教师的关联,表明课程是由哪位教师负责。这里的教师编号就是所有用户编号中的一个,只是

名字不同而已。

- 用户名和密码：是必需的，作为用户登录的入口，其中用户名要求是唯一的，即在数据库表里是互不相同的。
- 角色编号：是角色表的外键，在这里是为了表明用户的身份，明确用户属于哪类角色，它指向角色表的编号字段。
- 真实姓名、身份证号、联系电话、电子邮件，还有没有列举出来的字段如性别、地址等，都是作为用户信息的一个最基本的属性。其中，身份证号还有另外一个作用，就是当用户忘了密码时，可以通过出示身份证号获取丢失的密码。

根据系统的要求，管理员不能通过注册取得，所以在系统投入使用之前，用户表初始化了一条系统管理员的记录，用户名和密码都是 admin，其他信息由管理员登录后在后台自己填写，密码也由管理员在以后的使用中自己修改。

3. 栏目表（Column）设计

表 5-25 所示为栏目表，记录着系统的栏目信息。栏目表主要有两个作用，一是为了防止冗余，因为在一般情况下，在文章表（Article）中，每篇文章都应该标明属于哪个栏目，往往需要加上栏目名称的字段。当系统运行时间长了以后，文章的数量就会增加，导致存储的栏目信息过多，且冗余量大，所以在远程教育系统中把栏目单独抽出来作为一张表，而仅用栏目编号与文章表关联，就可以很好地解决这个问题。二是为了满足以后系统扩展，当需要修改栏目或者新增其他的栏目时，只需在栏目表修改即可。

表 5-25　栏目表

序号	字段含义	字段名称	类型	备　　注
1	栏目编号	ID	int	主码，需设置为自增长
2	栏目名称	ColumnName	varchar	

表 5-25 中的各字段说明如下。

- 栏目编号（ID）：是表的主键，唯一标识一个栏目；
- 栏目名称（ColumnName）：是栏目的名称。

栏目信息表也是系统的基础表，在系统投入使用之前就已经进行了数据的初始化，如表 5-26 所示。

表 5-26　栏目表初始化数据

栏目编号（ID）	栏目名称（ColumnName）	栏目编号（ID）	栏目名称（ColumnName）
1	系统简介	5	作业展示
2	培训公告	6	思考习题
3	学习简报	7	答疑解惑
4	友情链接		

4. 文章表（Article）设计

表 5-27 所示为文章表，首页中系统简介、培训公告、学习简报和答疑解惑栏目的文章

信息都存放在该表中。由上面的逻辑结构可知,文章与发表人的关系为多对一,也就是说一篇文章只能由某个人发表,一个人可以发表多篇文章。文章与栏目的关系也是多对一的关系,也就是一篇文章只能属于某个栏目,一个栏目可以包含多篇文章。

表 5-27 文章表 Article

序号	字段含义	字段名称	类型	备 注
1	文章编号	ID	int	主键,需设置为自增长
2	用户编号	UserID	int	与用户表关联
3	栏目编号	ColumnID	int	与栏目表关联
4	文章标题	Title	varchar	
5	文章内容	Content	varchar	
6	发表时间	PublishTime	datetime	

表 5-27 中的各字段说明如下。

- 文章编号(ID):为文章表的主键,唯一标识一篇文章。
- 用户编号(UserID):为用户表的外键,指向用户表的编号字段,表明文章是谁发布的。只有管理员有发表文章的权限。
- 栏目编号(ColumnID):为栏目表的外键,指向栏目表的编号字段,表明文章属于哪个栏目。
- 文章标题(Title)、文章内容(Content)和发表时间(PublishTime)是文章的基本信息。

5. 课程表(Course)的设计

表 5-28 所示为课程表,记录着系统中所有课程的信息。由上面的逻辑结构可知,课程与教师的关系为多对一,也就是说一门课程只能由某个教师负责,一个教师可以负责多门课程。

表 5-28 课程表 Course

序号	字段含义	字段名称	类型	备 注
1	课程编号	ID	int	主键,需设置为自增长
2	课程名称	CourseName	varchar	
3	教师编号	TeacherID	int	与用户表关联

表 5-28 中的各字段说明如下。

- 课程编号(ID):为课程表的主键,唯一标识一门课程。
- 课程名称(CourseName):为课程的名字,该字段的宽度是 50,创建时,应控制课程名称的长度。
- 教师编号(TeacherID):为用户表的外键,指向用户表的编号字段。任课教师具有操作该课程的权限。

6. 习题表(HomeWork)的设计

表 5-29 所示为习题表，记录着教师发布的作业和思考习题的相关信息。由逻辑结构可知，习题与栏目是多对一的关系，习题与课程是多对一的关系，习题与教师也是多对一的关系。

表 5-29　习题表 HomeWork

序号	字段含义	字段名称	类型	备　注
1	习题编号	ID	int	主键，需设置为自增长
2	栏目编号	ColumnID	int	与栏目表关联
3	课程编号	CourseID	int	与课程表关联
4	教师编号	TeacherID	int	与用户表关联
5	习题题目	Title	varchar	
6	习题内容	Content	varchar	
7	发表时间	PublishTime	datetime	

表 5-29 中的各字段说明如下。

- 习题编号(ID)：为习题表的主键，唯一标识一道习题。
- 栏目编号(ColumnID)：为习题所属的栏目，是栏目表的外键。在该系统中，值只可能为作业和思考习题栏目编号。
- 课程编号(CourseID)：为习题所属的课程，是课程表的外键，指向课程表中课程编号。
- 教师编号(TeacherID)：为用户表的外键，指向用户表的编号字段，表明习题是由谁发布的(注：根据前面的分析，课程与教师是多对一的关系，也就是说一门课程只能由一个教师负责，由此可知，习题所属的课程确定了，教师也就确定了。所以在该表中，教师编号为冗余字段。但考虑到性能因素，加上了该字段)。
- 习题题目(Title)、习题内容(Content)和发表时间(PublishTime)为习题的基本信息。

7. 答题信息表(Answer)的设计

表 5-30 所示为答题信息表，记录着学员答题的相关信息。该表关系比较复杂，一方面是学员答题的内容，另一方面是教师批改的相关信息，这两方面不是同时进行的。只有学员先做作业或者做思考习题，提交后，教师才能批改。

表 5-30　答题信息表 Answer

序号	字段含义	字段名称	类型	备　注
1	答题编号	ID	int	主键，需设置为自增长
2	习题编号	HomeworkID	int	与习题表关联
3	学员编号	StudentID	int	与用户表关联

续表

序号	字段含义	字段名称	类型	备　注
4	答题时间	AnswerTime	datetime	
5	答题内容	Content	varchar	
6	教师评语	TeacherComment	varchar	
7	教师评分	HomeworkS	float	

表 5-30 中的各字段说明如下。

- 答题编号(ID)：为答题表的主键，唯一标识一次答题信息。
- 习题编号(HomeworkID)：为所答习题编号，是习题表的外键。
- 学员编号(StudentID)：为回答习题的学员，是用户表的外键，指向用户表中用户编号。
- 答题时间(AnswerTime)、答题内容(Content)为答题的基本信息。
- 教师评语(TeacherComment)、教师评分(HomeworkS)为教师批改的相关信息。

8. 友情链接表(FriendshipLink)的设计

表 5-31 所示为友情链接表，记录着友情链接的相关信息。该表跟其他表没有关联，由管理员负责维护。

表 5-31　友情链接表 FriendshipLink

序号	字段含义	字段名称	类型	备　注
1	链接编号	ID	int	主键，需设置为自增长
2	链接名称	LinkTitle	varchar	
3	链接地址	LinkUrl	varchar	
4	添加时间	AddTime	datetime	

表 5-31 中的各字段说明如下。

- 链接编号(ID)：为友情链接表的主键，唯一标识一个友情链接。
- 链接名称(LinkTitle)、链接地址(LinkUrl)是友情链接的基本信息。
- 添加时间(AddTime)：为管理员添加友情链接的时间。

5.3 逻辑分层

这一章节主要介绍开发应用程序时采用的逻辑分层思想，以及三层架构在开发示例中的应用。

从历史角度来说，有两种力量相互结合催生了逻辑分层方法。第一种是对可伸缩性的需求。随着 Internet 的发展及 Web 为越来越多的最终用户所使用，企业意识到他们可以将企业系统推向个人客户，将许多从前由内部系统(呼叫中心员工)所做的工作从公司

转移到 Web 上。例如，在 1980 年，客户必须打电话给运输公司，来向客户服务代表询问某件包裹到了哪里。客户服务代表会询问跟踪号码，然后使用内部软件系统来发现包裹的位置。而到了 2005 年，该客户只需将所选 Web 浏览器指向运输公司的 Web 站点，并输入跟踪号码。使用相同的后端算法搜索同一后端数据存储系统，而不同的是，现在信息直接由客户输入而不是通过内部员工间接输入。但是，企业系统覆盖范围的扩大势必伴随着相应的成本，从前只有几百名用户(客户服务代表)的内部系统，目前却拥有几十万的潜在用户(客户)。这时会遇到一个瓶颈问题，大多数数据库服务器可以支持几百个并发的连接，然而并发连接如果达到几十万，很快就会使数据库陷于瘫痪。

但结果是，人们发现了这些并发连接的一个有趣特性：对于大多数客户机/服务器应用环境而言，针对数据库所建立的连接大多数时间(95%以上)都处于空闲状态，来等待要针对数据库所执行的请求。也就是说，瓶颈在于连接的数量，而不在于所执行的任务。这就说明，要增加数据库的可伸缩性，需要以某种方式增加通过这些连接所完成的工作量。因此，要创建一个过渡层，客户机通过连接该过渡层以多工方式向数据库传输请求。简单地说，如果数据库只能支持 100 个连接，而每个客户机连接被用去 1%的时间，那么可以这样来增加数据库的可伸缩性：将 100 个客户机连接到中间服务器，这个中间服务器仅使用一个连接(总共 100%的时间，每个客户端占 1%)来对数据库进行操作，可伸缩性则增加了 100 倍。

尽管如此，有多少应用环境真正需要这种可伸缩性呢？当然，在将企业应用环境带到最终客户群面前的情况下，这种可伸缩性可能是必要的。但许多应用环境(无论是否基于 Web)仍然是完全部署在内部的，这种情况下只有不到 100(有时少于 10)个客户机将同时访问系统。在内部小用户群应用环境中还需要逻辑分层方法吗？

这时安全因素发挥了作用。对于运行在最终用户计算机上的应用程序(基于 Web 或“富客户机”)，任何系统管理员或安全顾问都不可能推荐将包含关键任务数据的数据库直接放在防火墙的后面(例如，从在安全范围之外运行的计算机直接访问)。如果加入一个中间计算机并将另一个防火墙置于其后，可以创建通常称为隔离区或 DMZ 的区域，在其中可以进一步限制对数据库的访问。这样一个 DMZ 极大地巩固了安全基础结构，降低了成功渗透的可能性。这不仅可以防止数据被窃取，而且还有助于保护服务器(及剩下的应用程序或系统)免受成功的拒绝服务攻击。

使许多大型系统所有者青睐逻辑分层系统的另一个因素是部署，即物理地将软件放入客户机所能访问的计算机的过程。在传统的客户机/服务器环境中，业务逻辑与表现逻辑和数据访问逻辑交织在一起，使编程人员感觉非常不舒服：每次需要新的更新时(例如需要更改企业对数据的处理方式或增加数据的新视图)，位于用户桌面的“胖客户机”就必须更换并/或加入新代码。这就意味着，至少在这个时候，有人(通常是开发人员或系统管理员，两者中级别较低的一个)需要在各计算机之间奔走，来安装新代码。或者是要求用户从网络下载最新的代码。当然，大多数的用户不是忽略了这一点就是不能正确操作。这两种情况都不能使人们对频繁发布这种明智之举真正产生信任。而且部署是需要时间的，在这段时间内，系统必须暂停工作，以避免因混合版本的应用程序匆忙操作数据库，而造成的任何类型的语义数据损坏。

这一部署因素在很大程度上影响着人们对逻辑分层模型，特别是对基于 Web 的应用程序的青睐程度。现在，人们无需再将代码部署到各用户桌面，而可以将其部署到（单个）Web 服务器，最终用户的浏览器只需去选取这些更改，而这并不需要任何进一步的工作。就其本身而言，部署并不是铺开逻辑分层系统的原因。在传统客户机/服务器应用环境时代所无法获得的一些备选方案现在已经加入可能部署方案清单中，这其中包括“无触式部署”（在 .NET 1. x 中）和 ClickOnce（在 .NET 2. 0 中），更不用说人们对 AJAX 和各种混合式结构日益浓厚的兴趣。事实上，发布能在启动时进行自我更新的富客户机应用程序（如在 iTunes 软件管理器、Windows Media Player，甚至是流行的 .NET 开发工具 Reflector 中所看到的）已变得越来越普遍。

采用逻辑分层的第三个原因经常会被人们所提到，但往往却不能真正实现，即中间物理层可以成为一些业务逻辑的集中点，这些逻辑可以由多个表现物理层访问，却不能为它们所知。关于这一点的典型例子就是组合式内联网/外联网应用环境，其中内部员工使用 WinForms（很快将升级为 Windows Presentation Foundation）应用程序来访问中间层系统，中间层系统进而再访问数据库，而外部用户（合作伙伴和/或客户）使用 ASP. NET 或者可能是基于 SharePoint 的 Web 站点来进行同样的操作：访问中间层，中间层进而再访问数据库。

软件设计逻辑分层在软件体系结构领域已被普遍接受。“逻辑分层”是对软件的逻辑分割，是在开发人员级对各关注点的基础分割，这样可以更加容易地划分系统的职责。使用逻辑层有助于构筑可以分解为子任务组（每组子任务处于某个抽象级别）的应用程序。换句话说，它是对各关注点的典型分割方式：将企业系统中所涉及的各种任务（包括检索数据、存储数据、针对数据执行业务规则、显示数据和搜集输入值等）分割为组件或子节，以便能够更加轻松地跟踪在何时何处发生了什么。最为常见的方式是将任务分为“表现”、“逻辑”和“数据访问”（逻辑）层。

而另一方面，“物理层”是硬件（通常是某种形式的计算机）的物理分层，系统的一部分或全部可以运行于其上。传统的客户机/服务器计算（即写入可以针对运行于单独服务器上的数据库执行 SQL 语句的程序）是一个由两个物理层构成的系统。我们每天都享用的万维网也是在两（物理）层方法的基础上构建的，其中一个物理层（即客户机）位于某人的家中或办公室内，并远程访问位于某处服务器机房中的另一个物理层。这样的例子数不胜数。

相信大家以前都接触过开发软件时采用的分层模型。为什么要使用分层体系结构来构建系统呢？这是人们对于软件开发的一种理念，即当我们面对新项目时往往将系统明确地分为三个物理层次：表现层、业务逻辑层和数据访问或资源层。

为什么要使用分层体系结构来构建系统？这是人们对于软件开发的一种理念，即当我们面对新项目时往往将系统明确地分为三个物理层次：表现层，用以处理所有用户输入和数据显示问题；业务逻辑层，存放所有“业务逻辑”；数据访问或资源层，存放用于检索、修改或存储数据的所有代码。

5.3.1　三层体系架构

1. 表示层(USL)

主要表示 Web 方式，也可以表示成 WinForm 方式。基于 Web 应用程序中的“表现”逻辑层是 Web 浏览器。浏览器将要显示的 HTML 通常必须从服务器内所运行的某种形式(ASP 或 ASP.NET)的代码组件中生成并被发送给浏览器。如果逻辑层相当强大和完善，无论表现层如何定义和更改，逻辑层都能完善地提供服务。

2. 业务逻辑层(BLL)

主要是针对具体问题的操作，也可以理解成对数据层的操作，对数据业务逻辑处理。“数据访问”逻辑层用于访问和处理数据的命令(即 SQL)必须从数据库层之外被生成和发送。如果说数据层是积木，那逻辑层就是对这些积木的搭建。

3. 数据访问层(DAL)

主要是对原始数据(数据库或者文本文件等存放数据的形式)的操作层，而不是指原始数据。也就是说，是对数据的操作，而不是数据库，具体为业务逻辑层或表示层提供数据服务。

5.3.2　具体区分

(1) 表示层：主要对用户的请求接受，以及数据的返回，为客户端提供应用程序的访问。

(2) 业务逻辑层：主要负责对数据层的操作，也就是说把一些数据层的操作进行组合。

(3) 数据访问层：主要看数据层里面有没有包含逻辑处理，实际上它的各个函数主要完成各个对数据文件的操作，而不必管其他操作。

三层结构是一种严格分层方法，即数据访问层只能被业务逻辑层访问，业务逻辑层只能被表示层访问，用户通过表示层将请求传送给业务逻辑层，业务逻辑层完成相关业务规则和逻辑，并通过数据访问层访问数据库获得数据，然后按照相反的顺序依次返回将数据显示在表示层。

当然，在三层结构基础上可以扩展成 N 层，可以作为了解。对于大多数人来说，N 层应用程序就是被分成多个独立的逻辑部分的应用程序。最常见的选择是分为三个部分：表示、业务逻辑和数据，当然还可能存在其他的划分方法。N 层应用程序最初是为了解决与传统的客户机/服务器应用程序相关的问题而出现的，但是，随着 Web 时代的到来，这一体系结构开始成为新开发项目的主流。

将应用程序分解成多个逻辑部分是很有用的。将一个大软件分成几个小的部分会更利于软件的构建、重复利用和修改，对适应不同的技术或不同的业务组织也很有帮助。同时，还有一些综合因素需要考虑。虽然模块化和重复使用性很有效，但它们可能会导致应用程序不能像使用其他方法那样安全、易管理和快速。

5.3.3 举例说明三层结构的应用

1. 实现效果

培训公告类似于新闻发布功能，由管理员在后台增加、删除和修改。首页显示内容，用户单击标题链接进入培训公告详细信息页面。

2. 数据访问层

培训公告模块的数据访问层为 DAL 文件夹下的 DBAccess. cs、SqlStringFormat. cs 和 GetSafeData. cs。该数据访问层提供了培训公告模块业务逻辑层所使用的对培训公告的增加、删除和修改等的方法。以下是 DBAccess. cs 的代码清单，如程序 5-1 所示。

程序 5-1 DBAccess. cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.ComponentModel;
using System.Collections;
using System.Diagnostics;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
namespace EDUNEMP.DAL
{
    //类,用于数据访问的类
    public class DBAccess: IDisposable
    {
        //保护变量,数据库连接
        protected SqlConnection Connection;
        //保护变量,数据库连接串
        protected String ConnectionString;
        //构造函数
        //<param name="DatabaseConnectionString">数据库连接串</param>
        public DBAccess()
        {
            ConnectionString=ConfigurationManager.AppSettings["NEMPOffice"];
        }
        //构造函数
        //<param name="pDatabaseConnectionString">数据库连接串</param>
        public DBAccess(string pDatabaseConnectionString)
        {
            ConnectionString=pDatabaseConnectionString;
        }
        //析构函数,释放非托管资源
```

```
~DBAccess()
{
    try
    {
        if (Connection != null)
            Connection.Close();
    }
    catch{}
    try
    {
        Dispose();
    }
    catch{}
}
/// <summary>
/// 保护方法,打开数据库连接
/// < /summary>
protected void Open()
{
    if (Connection==null)
    {
        Connection=new SqlConnection(ConnectionString);
    }
    if (Connection.State.Equals(ConnectionState.Closed))
    {
        Connection.Open();
    }
}
/// <summary>
/// 公有方法,关闭数据库连接
/// </summary>
public void Close()
{
    if (Connection != null)
        Connection.Close();
}
/// <summary>
/// 公有方法,释放资源
/// </summary>
public void Dispose()
{
    // 确保连接被关闭
    if (Connection != null)
```

```
                {
                    Connection.Dispose();
                    Connection=null;
                }
            }
        }
```

GetDataSet 方法首先利用 DBAccess 类提供的 Open 方法创建了数据库连接，然后创建了数据库命令，并指定数据库命令执行 SqlString 语句。本方法的参数 SqlString，本方法返回 dataset 数据集，如程序 5-2 所示。

程序 5-2　DBAccess. cs

```
/// <summary>
/// 公有方法,获取数据,返回一个 DataSet
/// </summary>
/// <param name="SqlString">Sql 语句</param>
/// <returns>DataSet</returns>
public DataSet GetDataSet(String SqlString)
{
    Open();
    SqlDataAdapter adapter=new SqlDataAdapter(SqlString,Connection);
    DataSet dataset=new DataSet();
    adapter.Fill(dataset);
    Close();
    return dataset;
}
/// <summary>
/// 公有方法,获取数据,返回一个 DataRow
/// </summary>
/// <param name="SqlString">Sql 语句</param>
/// <returns>DataRow</returns>
public DataRow GetDataRow(String SqlString)
{
    DataSet dataset=GetDataSet(SqlString);
    dataset.CaseSensitive=false;
    if (dataset.Tables[0].Rows.Count>0)
    {
        return dataset.Tables[0].Rows[0];
    }
    else
    {
        return null;
    }
}
```

```
/// <summary>
/// 公有方法,执行一组 Sql 语句
/// </summary>
/// <param name="SqlStrings">Sql 语句组</param>
/// <returns>是否成功</returns>
public bool ExecuteSQL(ArrayList SqlStrings)
{
    bool success=true;
    Open();
    SqlCommand cmd=new SqlCommand();
    SqlTransaction trans=Connection.BeginTransaction();
    cmd.Connection=Connection;
    cmd.Transaction=trans;
    try
    {
        foreach (String str in SqlStrings)
        {
            cmd.CommandText=str;
            cmd.ExecuteNonQuery();
        }
        trans.Commit();
    }
    catch
    {
        success=false;
        trans.Rollback();
    }
    finally
    {
        Close();
    }
    return success;
}
```

增加新文章的公有方法,将页面上发表的新文章写到数据库中,如程序 5-3 所示。

程序 5-3　DBAccess. cs

```
/// <summary>
/// 公有方法,在一个数据表中插入一条记录
/// </summary>
/// <param name="TableName">表名</param>
/// <param name="Cols">哈西表,键值为字段名,值为字段值</param>
/// <returns>是否成功</returns>
```

```
public bool Insert(String TableName,Hashtable Cols)
{
    int Count=0;
    if (Cols.Count<=0)
    {    return true;}
    String Fields=" (";
    String Values=" Values(";
    foreach(DictionaryEntry item in Cols)
    {
        if (Count != 0)
        {
            Fields+=",";
            Values+=",";
        }
        Fields+=item.Key.ToString();
        Values+=item.Value;
        Count++;
    }
    Fields+=")";
    Values+=")";
    String SqlString="Insert into"+TableName+Fields+Values;
    return Convert.ToBoolean(ExecuteSQL(SqlString));
}
```

根据参数 Where 设定的条件对数据库中的相关记录进行修改，在培训公告模块中是对培训公告文章的详细信息进行修改，如程序 5-4 所示。

程序 5-4　DBAccess. cs

```
/// <summary>
/// 公有方法,更新一个数据表
/// </summary>
/// <param name="TableName">表名</param>
/// <param name="Cols">哈西表,键值为字段名,值为字段值</param>
/// <param name="Where"> Where 子句</param>
/// <returns>是否成功</returns>
public bool Update(String TableName,Hashtable Cols,String Where)
{
    int Count=0;
    if (Cols.Count<=0)
    { return true;}
    String Fields=" ";
    foreach(DictionaryEntry item in Cols)
    {
        if (Count != 0)
```

```
            { Fields+=","; }
            Fields+=item.Key.ToString();
            Fields+="=";
            Fields+=item.Value.ToString();
            Count++;
        }
        Fields+=" ";
        String SqlString="Update"+TableName+" Set "+Fields+Where;
        return Convert.ToBoolean(ExecuteSQL(SqlString));
    }
```

以下是 SqlStringFormat.cs 类的代码清单，如程序 5-5 所示。

程序 5-5　SqlStringFormat.cs

```
using System;
namespace EDUNEMP.DAL
{
   public class SqlStringFormat
    {
        /// <summary>
        /// 公有静态方法,将文本转换成适合在 Sql 语句里使用的字符串
        /// </summary>
        /// <returns>转换后文本</returns>
        public static String GetQuotedString(String pStr)
        {
            return ("'"+pStr.Replace("'", "''")+"'");
        }
    }
}
```

这个类提供了一个公有静态方法，将参数 pStr 的文本转换成适合在 Sql 中使用的字符串。

以下是 GetSafeData.cs 类的代码清单，如程序 5-6 所示。

程序 5-6　GetSafeData.cs

```
using System;
using System.Data;
namespace EDUNEMP.DAL
{
    public class GetSafeData
    {
        #region DataRow
        /// <summary>
```

```
        /// 从一个 DataRow 中,安全地得到列 colname 中的值。值为字符串类型
        /// </summary>
        /// <param name="row">数据行对象</param>
        /// <param name="colname">列名</param>
        /// <returns>如果值存在,返回;否则,返回 System.String.Empty</returns>
        public static string ValidateDataRow_S(DataRow row, string colname)
        {
            if (row[colname] != DBNull.Value)
                return row[colname].ToString();
            else
                return System.String.Empty;
        }
        /// <summary>
        /// 从一个 DataRow 中,安全地得到列 colname 中的值。值为整数类型
        /// </summary>
        /// <param name="row">数据行对象</param>
        /// <param name="colname">列名</param>
        /// <returns>如果值存在,返回;否则返回 system.Int32.MinValue
        ///</returns>
        public static int ValidateDataRow_N(DataRow row, string colname)
        {
            if (row[colname] != DBNull.Value)
                return Convert.ToInt32(row[colname]);
            else
                return 0;
        }
        /// <summary>
        /// 从一个 DataRow 中,安全地得到列 colname 中的值。值为浮点数类型
        /// </summary>
        /// <param name="row">数据行对象</param>
        /// <param name="colname">列名</param>
        /// <returns>如果值存在,返回;否则返回 System.Double.MinValue
        ///</returns>
        public static double ValidateDataRow_F(DataRow row, string colname)
        {
            if (row[colname] != DBNull.Value)
                return Convert.ToDouble(row[colname]);
            else
                return System.Double.MinValue;
        }
    /// <summary>
    /// 从一个 DataRow 中,安全地得到列 colname 中的值。值为时间类型
    /// </summary>
    /// <param name="row">数据行对象</param>
```

```
        /// <param name="colname">列名</param>
        /// <returns>如果值存在,返回;否则返回 system.DateTime.MinValue;
        ///</returns>
            public static DateTime ValidateDataRow_T(DataRow row, string colname)
            {
                if (row[colname] != DBNull.Value)
                    return Convert.ToDateTime(row[colname]);
                else
                    return System.DateTime.MinValue;
            }
            /// <summary>
            /// 从一个 DataRow 中,安全地得到列 colname 中的值。值为布尔类型
            /// </summary>
            /// <param name="row">数据行对象</param>
            /// <param name="colname">列名</param>
            /// <returns>如果值存在,返回;否则返回 false;</returns>
            public static Boolean ValidateDataRow_B(DataRow row, string colname)
            {
                if (row[colname] != DBNull.Value)
                    return Convert.ToBoolean(row[colname]);
                else
                    return false;
            }
            #endregion DataRow
        }
    }
```

这个类提供了 5 个公有静态方法,分别从参数 row 中安全地得到列 colname 中的值。值的类型分别为字符串类型、整数类型、浮点数类型、时间类型和布尔类型。

3. 业务逻辑层

这一层包括 Article.cs 实体业务类,提供了对文章进行增加、删除和修改的具体方法。还有显示文章列表管理页面的后台代码,及增加文章页面的后台代码和修改文章页面的后台代码。

以下是 Article.cs 类的代码清单,如程序 5-7 所示。

程序 5-7　Article.cs

```
using System;
using System.Collections;
using System.Data;
using System.Text;
using EDUNEMP.DAL;
namespace EDUNEMP.BLL
```

```
{
    public class Article
    {
        ///<Summary>
        /// 文章相关字段
        ///</Summary>
        #region 私有成员
        private int _id;
        private int _columnId;
        private int _userId;
        private string _title;
        private string _content;
        private DateTime _publishTime;
        private int _clickTimes;
        private string _userName;
        #endregion
        #region 属性
        public int ID
        {
            set  {  this._id=value;  }
            get  {  return this._id;   }
        }
        public int ColumnID
        {
            set  {  this._columnId=value; }
            get  {  return this._columnId;  }
        }
        public int UserID
        {
            set  { this._userId=value;    }
            get  { return this._userId;     }
        }
        public string Title
        {
            set  { this._title=value;    }
            get  { return this._title;    }
        }
        public string Content
        {
            set  { this._content=value;  }
            get   { return this._content;  }
        }
        public DateTime PublishTime
```

```
        {
            set  { this._publishTime=value;  }
            get  { return this._publishTime;   }
        }
        public int ClickTimes
        {
            set  { this._clickTimes=value;  }
            get  { return this._clickTimes;  }
        }
        public string UserName
        {
            set  { this._userName=value; }
            get   { return this._userName; }
        }
        #endregion 属性
        #region 方法
        /// <summary>
        /// 根据参数 ArticleID,获取文章详细信息
        /// </summary>
        public void LoadData(int ArticleID)
        {
            DBAccess db=new DBAccess();          //实例化一个 DBAccess 类
            string sql="";
            sql="Select Article.[ID],ColumnID,Title,Content,PublishTime,ClickTimes,
[User].RealName from Article,[User] where ColumnID="+ColumnID;
            sql+="  and  Article.UserID=User.ID";
            DataRow dr=db.GetDataRow(sql);
                                   //利用 DBAccess 类的 GetDataRow 方法查询用户数据
            //根据查询得到的数据,对成员赋值
            if (dr != null)
            {
                this._id=GetSafeData.ValidateDataRow_N(dr, "ID");
                this._userId=GetSafeData.ValidateDataRow_N(dr, "UserID");
                this._title=GetSafeData.ValidateDataRow_S(dr, "Title");
                this._content=GetSafeData.ValidateDataRow_S(dr, "Content");
                this._publishTime=GetSafeData.ValidateDataRow_T(dr, "PublishTime");
                this._clickTimes=GetSafeData.ValidateDataRow_N(dr, "ClickTimes");
                this._columnId=GetSafeData.ValidateDataRow_N(dr, "ColumnID");
                this._userName=GetSafeData.ValidateDataRow_S(dr, "RealName");
            }
        }
        ///<summary>
        ///栏目中文章数据源
```

```
        ///</summary>
        public static DataSet QueryArticle(int ColumnID)
        {
            string sql="";
            sql="Select Article.[ID],ColumnID,Title,Content,PublishTime,ClickTimes,
  [User].RealName from Article,[User] where ColumnID="+ColumnID+" and Article.UserID=
 [User].[ID]";
            DBAccess db=new DBAccess();
            return db.GetDataSet(sql);
        }
        ///<summary>
        ///根据文章 ID 删除该文章
        ///</summary>
        public void Delete(int ArticleID)
        {
            ArrayList sqls=new ArrayList();
            string sql="";
            sql="delete Article where ArticleID="+ArticleID;
            sqls.Add(sql);
            DBAccess db=new DBAccess();
            db.ExecuteSQL(sqls);
        }
        ///<summary>
        ///向数据库添加一个文章
        ///</summary>
        ///<param name="topicInfo">文章信息哈希表</param>
        public void Add(Hashtable articleInfo)
        {
            DBAccess db=new DBAccess();         //实例化一个 Database 类
            db.Insert("[Article]", articleInfo);
                                               //利用 Database 类的 Insert 方法,插入数据
        }
        ///<summary>
        ///修改文章内容
        ///</summary>
        public void Update(Hashtable newArticleInfo, int ArticleID)
        {
            DBAccess db=new DBAccess();
            string strCond="Where ArticleID="+ArticleID;
            db.Update("[Article]", newArticleInfo, strCond);
        }
        #endregion  方法
    }
}
```

本类提供了对文章进行增加、删除和修改的业务方法。由该层提供的方法去调用数据访问层中的方法,达到修改数据库中记录及页面显示效果的目的。

BulletinList.aspx.cs 类提供了显示文章列表管理页面的后台代码,如程序 5-8 所示。该页面显示某个栏目下的所有文章列表,并提供增加、删除和修改等方法的链接。

程序 5-8　BulletinList.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Text;
using EDUNEMP.BLL;
using EDUNEMP.DAL;
namespace EDUNEMP.Web
{
  public partial class BulletinList : System.Web.UI.Page
  {
    protected int ColumnID;
    protected int ID;
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            if (!CheckUser())
            {
                Response.Redirect("default.aspx");
            }
            else
            {
                ColumnID=Convert.ToInt32(Request.QueryString["ColumnID"]);
                Session["ColumnID"]=ColumnID;
                InitData();
            }
            this.DataBind();
        }
    }
```

```
#region Web 窗体设计器生成的代码
override protected void OnInit(EventArgs e)
{
    // CODEGEN: 该调用是 ASP.NET Web 窗体设计器所必需的
    InitializeComponent();
    base.OnInit(e);
}
/// <summary>
/// 设计器支持所需的方法,不要使用代码编辑器修改
/// 此方法的内容
/// </summary>
private void InitializeComponent()
{
}
#endregion
/// <summary>
/// 验证用户身份
/// </summary>
/// <returns></returns>
private bool CheckUser()
{
    if ((UserData)Session["current_user"]==null)
        return false;
    else
      if (Convert.ToInt32(((UserData)Session["current_user"]).RoleID)==1)
      {  return true; }
        else
        {  return false; }
}
/// <summary>
/// 按钮列单击事件
/// </summary>
protected void GV_RowCommand(object sender, GridViewCommandEventArgs e)
{
    int ID=-1;
    Article article=new Article();
    int articleId=Convert.ToInt32(e.CommandArgument);
    int count=GV.Rows.Count;
    switch (e.CommandName)
    {
          //单击列表中删除按钮的响应事件
          case "delete":
            article.LoadData(articleId);
```

```
                    Response.Write("<Script Language=JavaScript>alert('删除成功!');</
                    Script>");
                    article.Delete(articleId);
                    InitData();
                    break;
                //单击列表中修改按钮的响应事件
                case "update":
                  string transferString=string.Format("ArticleEdit.aspx? ArticleID={0}",
                  ArticleID.ToString());
                   Response.Redirect(transferString);
                    break;
                default:
                    break;
            }
        }
        /// <summary>
        /// 列表该栏目下所有文章
        /// </summary>
        private void InitData()
        {
            ColumnID=Convert.ToInt32(Request.QueryString["ColumnID"]);
            Session["ColumnID"]=ColumnID;
            DataSet ds=Article.QueryArticle(ColumnID);
            GV.DataSource=ds;
            GV.DataBind();
        }

        protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
        {
            if ((UserData)Session["current_user"]==null)
            {
                Response.Write("<Script Language=JavaScript>alert('请先登录,登录之后才
能发表文章和评论!');</Script>");
            }
            else
            {
                string transferString=string.Format("ArticleAdd.aspx? ColumnID={0}",
ColumnID.ToString());
               Response.Redirect(transferString);
            }
        }
        /// <summary>
        /// 翻页事件
```

```
        /// </summary>
        protected void GV_PageIndexChanging(object sender, GridViewPageEventArgs e)
        {
            GV.PageIndex=e.NewPageIndex;
            InitData();
        }
    }
}
```

ArticleAdd.aspx.cs 类是增加某个栏目下文章页面的后台代码，如程序 5-9 所示。该页面是用户在 BulletinList.aspx 页面单击发表新文章后进入的页面。把用户在页面各控件中添加的内容写到相关数据库的字段中。

程序 5-9　ArticleAdd.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using EDUNEMP.DAL;
using EDUNEMP.BLL;
using System.IO;
namespace EDUNEMP.Web
{
    /// <summary>
    /// ArticleAdd的摘要说明
    /// </summary>
public partial class ArticleAdd : System.Web.UI.Page
{
        protected int ColumnID;
        protected void Page_Load(object sender, System.EventArgs e)
        {
            if (!CheckUser())
        {
        Response.Write("<Script Language=JavaScript>alert('登录之后才能发表文章!');
</Script>");
        Response.Redirect("default.aspx");
        }
```

```
            else
             {
               Label1.Text= ((UserData)Session["current_user"]).UserName;
               }
          }
          private bool CheckUser()
            {
                   if ((UserData)Session["current_user"]==null)
                      { return false; }
                  else
                      { return true; }
           }
           protected void ButtonOK_Click(object sender, ImageClickEventArgs e)
           {
                  Article article=new Article();
               ColumnID=Convert.ToInt32(Request.QueryString["ColumnID"]);
               Session["ColumnID"]=ColumnID;
               Hashtable ht=new Hashtable();
               ht.Add("UserID", ((UserData)Session["current_user"]).UserID);
               ht.Add ("Title", SqlStringFormat.GetQuotedString (TextBoxTitle.Text.Trim
               ().ToString()));
               ht.Add("Content", SqlStringFormat.GetQuotedString(TB_Content.Text.Trim
               ().ToString()));
               ht.Add ("PublishTime", SqlStringFormat.GetQuotedString (System.DateTime.
               Now.ToString()));
               ht.Add("ColumnID", ColumnID);
               article.Add(ht);
              Response.Write("<Script Language=JavaScript>alert('成功发表文章!');</Script
              >");
               Response.Redirect("default.aspx");
             }
           }
}
```

ArticleEdit.aspx.cs 类是修改某个栏目下文章页面的后台代码，如程序 5-10 所示。该页面是用户在 Bulletin.aspx 页面单击修改进入的页面。把用户在页面各控件中修改后的内容写到数据库的相关字段中。

程序 5-10　ArticleEdit.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using EDUNEMP.DAL;
using EDUNEMP.BLL;
using System.IO;
namespace EDUNEMP.Web
{
    /// <summary>
    /// ArticleEdit 的摘要说明
    /// </summary>
    public partial class ArticleEdit : System.Web.UI.Page
    {
        protected int ColumnID;
        protected void Page_Load(object sender, System.EventArgs e)
        {
            if (!IsPostBack)
            {
                if (!CheckUser())
                {  Response.Redirect("default.aspx"); }
                else
                {
                    Label1.Text= ((UserData)Session["current_user"]).UserName;
                    InitData();
                }
            }
        }
        private bool CheckUser()
        {
          if ((UserData)Session["current_user"]==null)
          {
              Response.Write("<Script Language=JavaScript>alert('请登录!');
</Script>");
            return false;
          }
          return true;
        }
        private void InitData()
        {
            int articleID=Convert.ToInt32(Request.QueryString["ArticleID"]);
            Article article=new Article();
            article.LoadData(articleID);
```

```
                TextBoxTitle.Text=article.Title;
                Label1.Text=article.UserName.ToString();
                TB_Content.Text=article.Content.ToString();
            }
            /// <summary>
            /// 确定按钮单击事件
            /// </summary>
            protected void ButtonOK_Click(object sender, ImageClickEventArgs e)
            {
               Article article=new Article();
               int articleID=Convert.ToInt32(Request.QueryString["ArticleID"]);
               Hashtable ht=new Hashtable();
               ht.Add("Title", SqlStringFormat.GetQuotedString(TextBoxTitle.Text));
               ht.Add("Content", SqlStringFormat.GetQuotedString(TB_Content.Text));
               article.Update(ht, articleID);
               Response.Redirect("default.aspx");
                 Response.Write("<Script Language=JavaScript>alert('编辑成功!');
</Script>");
            }
        }
}
```

4. 用户表示层

在培训公告模块中的相关用户表示层页面有 BulletinList. aspx、ArticleAdd. aspx 和 ArticleEdit. aspx。

（1）BulletinList. aspx 页面是显示培训公告栏目下的所有文章类表并提供增加、删除和修改功能的链接。

BulletinList. aspx 页面的前台 Html 代码如程序 5-11 所示。

程序 5-11　BulletinList. aspx

```
<%@ Page Language="C#" MasterPageFile="MasterPage.master" AutoEventWireup="true" Co-
deFile="BulletinList.aspx.cs" Inherits="EDUNEMP.Web.BulletinList" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="
Server">
<table width="772" border="0" align="center" cellpadding="0" cellspacing="0" back-
ground="../Image/graybg.jpg">
            <tr>
                <td style="width: 772px; height: 25px; text-align: center;">
                    (本区域内显示在本栏目下查询到的文章列表)
                </td>
            </tr>
            <tr>
```

```
 < td valign = " top" align = " center " style = " width: 557px; height: 245px; text -
align: center;">
 <asp:GridView ID="GV" runat="server" AutoGenerateColumns="False" OnPageIndexChanging
 ="GV_PageIndexChanging" PageSize="20" OnRowCommand="GV_RowCommand" CellPadding="4"
 ForeColor="# 333333" Width="772px" DataKeyNames="ID" AllowPaging="True" Horizonta-
lAlign="Center">  HorizontalAlign="Center">
     <Columns>
        <asp:HyperLinkField DataNavigateUrlFields="ID"
                    DataNavigateUrlFormatString="Bulletin-info.aspx? ID={0}"
                  DataTextField="Title" HeaderText="标题" NavigateUrl="Bulletin-info.
 aspx">
                    <ControlStyle CssClass="link05" />
                    <ItemStyle HorizontalAlign="Left" Width="300px" />
 </asp:HyperLinkField>
 <asp:BoundField DataField="RealName" HeaderText="发布者" />
 <asp:BoundField DataField="PublishTime" HeaderText="发布时间" />
 <asp:TemplateField HeaderText="修改">
        <ItemTemplate>
          <asp:ImageButton ID="btnJH" runat="server" CommandName="update"
          CommandArgument='<%# Eval("ID")%>' ImageUrl="../Image/BtnEdit.gif" />
  </ItemTemplate>
 </asp:TemplateField>
 <asp:TemplateField HeaderText="删除">
          <ItemTemplate>
              <asp:ImageButton ID="btnSC" runat="server" CommandName="delete"
                   OnClientClick="return confirm('您确认将该文章删除吗?');"
                   CommandArgument='<%# Eval("ID") %>' ImageUrl="../Image/delete.jpg"
                   Height="22px" Width="45px" />
          </ItemTemplate>
   </asp:TemplateField>
   </Columns>
   <FooterStyle BackColor="# 5D7B9D" Font-Bold="True" ForeColor="White" />
   <RowStyle BackColor="# F7F6F3" ForeColor="# 333333" HorizontalAlign="Center" />
   <PagerStyle BackColor="# EBEBEC" HorizontalAlign="Center" />
   <HeaderStyle BackColor="# EBEBEC" Font-Bold="True" />
   <AlternatingRowStyle BackColor="White" ForeColor="Black" />
   </asp:GridView>   
              </td>
         </tr>
    <tr>
         <td align="center" style="height: 22px; width: 772px;">
           <asp:ImageButton ID="ImageButton1" runat="server"
              ImageUrl="../Image/SubmitArticle.jpg" OnClick="ImageButton1_Click" />
       </td>
    </tr>
```

```
    </table>
</asp:Content>
```

(2) ArticleAdd.aspx 页面是提供用户增加新文章的页面,具体前台页面代码见下一节培训公告管理模块的介绍。

(3) ArticleEdit.aspx 页面是提供用户修改某个文章的页面,具体前台页面代码见下一节培训公告管理模块的介绍。

5.4　功能模块介绍

通过前三节的介绍,已经对模块的功能、数据库及三层结构有了简单了解,下面来看一下各个模块的具体实现。逻辑层和数据层在前面三层结构中已经详细介绍过,本节会涉及到这两层中的方法,在这里只作简单的说明。

5.4.1　首页模块

图 5-6 所示是我们设计出的首页,设计中实现的众多功能大部分都能在首页中得到体现。



图 5-6　首页(Index.aspx、Index.aspx.cs)

从图 5-6 中可以看出，首页的结构非常简单，明显地可以看出被分割出的各个子单元，从上到下分别是网站标致(Logo)、导航栏的连接(系统首页、系统简介、培训公告、学习简报、答疑解惑、作业展示、思考习题和友情链接)、登录入口、学习简报栏目、答疑解惑栏目、友情链接栏目、系统简介栏目、培训公告栏目、作业展示栏目和思考习题栏目。

从图 5-6 中还可以看到，首页中的某些栏目只是简单的文字链接，如友情链接等。其余栏目如学习简报栏目、答疑解惑栏目、系统简介栏目、培训公告栏目、作业展示栏目和思考习题栏目等是纯文字文章列表。下面将分别介绍以上栏目的实现方法。

首先介绍系统简介栏目，如图 5-7 所示。

系统简介

广义上来说，远程教育是指将课件传送给外地的一处或多处学员的教育。依照这个定义，远程教育的历史可以追溯到19世纪30年代开始的商业函授课程。但是随着先进的信息技术，特别是Internet的出现，远程教育的特征发生了深刻的变化。的确，技术的迅速发展不断向远程教育的传统定义提出了挑战。结果，各种项目，技术支持，学员和媒体都大量使用“远程教育”这一术语。

远程教育是指通过音频、视频（直播或录像）及包括实时和非实时在内的计算机技术把培训课程传送到远处（校园外）的教育。这一远程教育的定义不包括专门在校园内及只通过书面形式进行的课程（尽管远程教育中也包括一些在校的指导或考试，其中一些指导 …… [查看全文]

图 5-7　系统简介栏目

要实现这样的效果，其前台代码片段如程序 5-12 和程序 5-13 所示。

程序 5-12　首页中的系统简介栏目（Index. aspx）

```
…
 <table width="530" border="0" cellpadding="0" cellspacing="0" class="bd">
   <asp:Label ID="SystemIntro" runat="server"></asp:Label>
 </table>
…
```

程序 5-13　首页中的系统简介栏目后台代码（Index. aspx. cs）

```
protected string modelnamextjj;
protected void Page_Load(object sender, EventArgs e)
{
    modelnamextjj=this.Page.Server.UrlEncode(">系统简介");
    StringBuilder sb1=new StringBuilder();
    DataTable IntroTable=new DataTable();
    IntroTable=IndexObject.GetSystemIntro();
    string systemintro=IntroTable.Rows[0]["Intro"].ToString();
    sb1.Append("        ").Append(system-
    intro.ToString().Substring(0, 296)).Replace("\r\n",
    "<br>        ").Append("<a  target='_
    blank' class='link02'
```

```
            href='SystemIntro.aspx?").Append("modelname=").Append(modelnamextjj).Append
            ("'>").Append("          
                [查看全文]").Append("</a>");
        SystemIntro.Text=sbl.ToString();
    }
```

通过调用 IndexObject 的方法 GetSystemIntro 来获得系统简介的内容。GetSystem-Intro 方法所涉及的 SQL 语句如下所示。

```
select Article.Content as Intro
from [Column] inner join Article on [Column].ID=Article.ColumnID
where [Column].ColumnName='系统简介'
```

下面再实现另一个栏目——培训公告栏目，如图 5-8 所示。

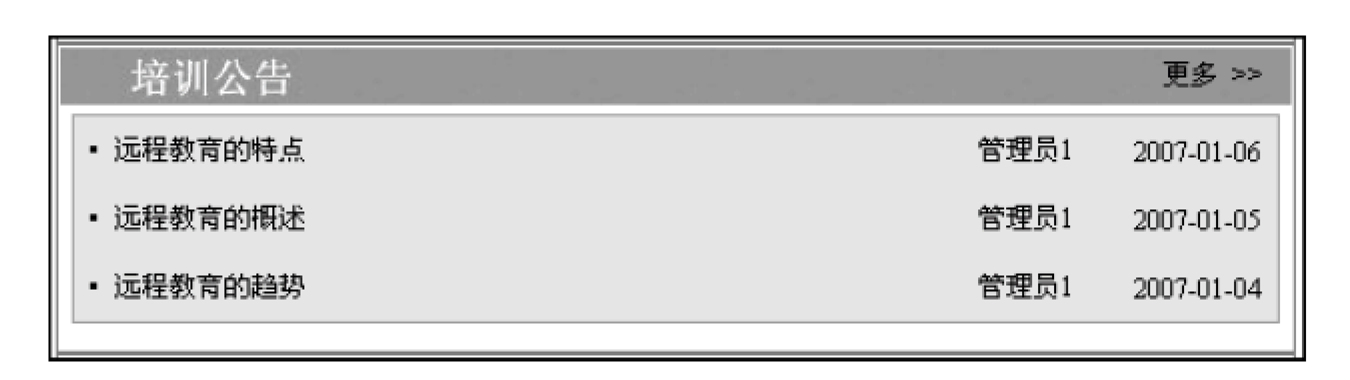


图 5-8　培训公告栏目

要实现这样的效果，其前台代码片段如程序 5-14 和程序 5-15 所示。

程序 5-14　首页中的培训公告栏目（Index. aspx）

```
<table width="540" height="25" border="0" cellpadding="0" cellspacing="0"
background="image/backpic.jpg">
    <tr>
      <td width="20" style="height: 25px"></td>
        <td style="height: 25px"><img src="Image/EDU/pxgg.jpg" width="95" height="25"
              style="margin-right:360px;"></td>
          <td width="75" style="height: 25px">
            <span class="link01"><a href="pxgg.aspx?modelname=<%#modelnamepxgg%>"
class="link01">
                    更多 &gt;&gt;</a></span></td></tr>
  </table>
  …
  <table width="530" border="0" cellpadding="0" cellspacing="0" class="bd">
      <asp:Label ID="pxgg" runat="server"></asp:Label>
  </table>
```

程序 5-15　首页中的培训公告栏目后台代码（Index. aspx. cs）

```
protected string modelnamepxgg;
protected void Page_Load(object sender, EventArgs e)
{
        modelnamepxgg=this.Page.Server.UrlEncode("> 培训公告");
        StringBuilder sb2=new StringBuilder();
        DataTable PxggTable=new DataTable();
        PxggTable=IndexObject.GetPxgg();
        for(int i=0;i<PxggTable.Rows.Count;i++)
          {
          sb2.Append("<tr>");
          sb2.Append("<td height='25' class='link01'><DIV
              align='left'>").Append(" ").Append(". ").Append(" ").Append("
              <A
              target='_blank' class='link01'
               href='pxgg-info.aspx?ArticleID=").Append(PxggTable.Rows[i]["Arti-
               cleID"].ToString()).Append("&").Append("modelname=").Append(model-
               namepxgg).Append("'>").Append(PxggTable.Rows[i]["ArticleTitle"].ToS-
               tring()).Append("</A></DIV></td>").Append("<td width='80' height='
               25'
              class='link01'>").Append(PxggTable.Rows[i]["UserName"].ToString()).Ap-
             pend("</td>").Append("<td width='70' height='30'
              class='link01'>").Append(PxggTable.Rows[i]["ArticlePublishTime"].ToS-
             tring()).Append("</td>");
          sb2.Append("</tr>");
         }
        pxgg.Text=sb2.ToString();
}
```

通过调用 IndexObject 的方法 GetPxgg 来获得培训公告的 ID 号、标题、发布者和发布时间，按照发布时间的先后顺序在页面上显示最新发布的 5 篇培训公告。当用户单击培训公告的标题时进入培训公告的文章浏览页面，当单击“更多”时进入培训公告的栏目页面（显示所有的培训公告列表）。GetPxgg 方法所涉及的 SQL 语句如下所示。

```
select distinct top 5 Article.ID as ArticleID,Article.Title as ArticleTitle,[User].Real-
Name
      as UserName,convert(varchar(10), Article.PublishTime,120)  as ArticlePublishTime
    from [Column] inner join Article on [Column].ID=Article.ColumnID
            inner join [User] on Article.UserID=[User].ID
    where [Column].ColumnName='培训公告'
    order by ArticlePublishTime desc
```

接下来实现另一个栏目——作业展示栏目，如图 5-9 所示。

要实现这样的效果，其前台代码片段如程序 5-16 和程序 5-17 所示。

程序 5-16　首页中的作业展示栏目（Index. aspx）

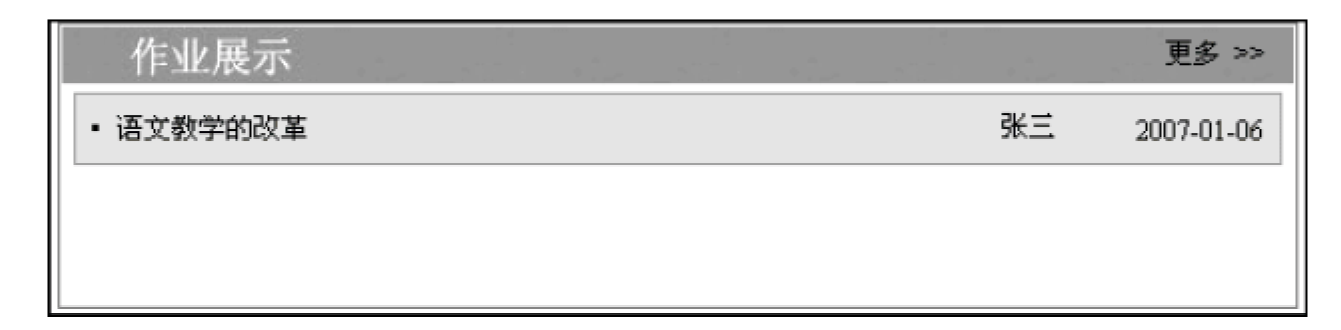


图 5-9　作业展示栏目

```
<table width="540" height="25" border="0" cellpadding="0" cellspacing="0" background=
"image/backpic.jpg">
   <tr>
    <td width="20"></td>
    <td><img src="Image/EDU/zyzs.jpg" width="95" height="25" style="margin-right:
360px;"></td>
    <td width="75">
      <span class="link01"><a href="zyzs.aspx?modelname=<%#modelnamezyzs%>" class
="link01">更多 &gt;&gt;</a></span></td>
   </tr>
</table>
<table width="530" border="0" cellpadding="0" cellspacing="0" class="bd">
  <asp:Label ID="zyzs" runat="server"></asp:Label>
</table>
```

程序 5-17　首页中的作业展示栏目后台代码（Index.aspx.cs）

```
protected string modelnamezyzs;
protected void Page_Load(object sender, EventArgs e)
{
    modelnamezyzs=this.Page.Server.UrlEncode(">作业展示");
    StringBuilder sb3=new StringBuilder();
    DataTable ZyzsTable=new DataTable();
    ZyzsTable=IndexObject.GetShowHomeWork();
    for (int i=0; i<ZyzsTable.Rows.Count; i++)
    {
       sb3.Append("<tr>");
       sb3.Append("<td height='25' class='link01'><DIV
                    align='left'>").Append(" ").Append(". ").Append(" ").
                    Append("<A target='_blank' class='link01'
                    href='zyzs-info.aspx?AnswerID=").Append(ZyzsTable.Rows[i]["An-
                   swerID"].ToString()).Append("&").Append("modelname=").Append
                    (modelnamezyzs).Append("'>").Append(ZyzsTable.Rows[i]["HomeWork-
                   Title"].ToString()).Append("</A></DIV></td>").Append("<td width
                    ='80' height='25'
```

```
                                class= 'link01'>").Append(ZyzsTable.Rows[i]["StudentName"].ToS-
                                tring()).Append("</td>").Append("<td width= '70' height= '30'
                                 class= 'link01'>").Append(ZyzsTable.Rows[i]["AnswerTime"].ToS-
                                tring()).Append("</td>");
        sb3.Append("</tr>");
      }
      zyzs.Text= sb3.ToString();
  }
```

通过调用 IndexObject 的方法 GetShowHomeWork 来获得学生回答作业的 ID 号、作业的标题、学生的姓名和回答作业的时间，按照分数的高低排序在页面上显示分数最高的 5 篇作业答卷。当用户单击作业的标题时进入作业的浏览页面，当单击“更多”时进入作业展示的栏目页面（显示所有的作业展示列表）。GetShowHomeWork 方法所涉及的 SQL 语句如下所示。

```
Select top 5  HomeWork.Title as HomeWorkTitle, [User].RealName as StudentName,
          convert(varchar(10), Answer.AnswerTime,120) as AnswerTime,Answer.ID as An-
swerID
from  HomeWork inner join Answer on  HomeWork.ID=Answer.HomeworkID
                  and
                  HomeWork.ColumnID=Answer.ColumnID
              inner join [Column] on  [Column].ID=Answer.ColumnID
              inner join [User] on [User].ID=Answer.StudentID
where [Column].ColumnName='作业展示'
order by HomeworkS desc
```

下面再来介绍一下友情链接栏目，如图 5-10 所示。

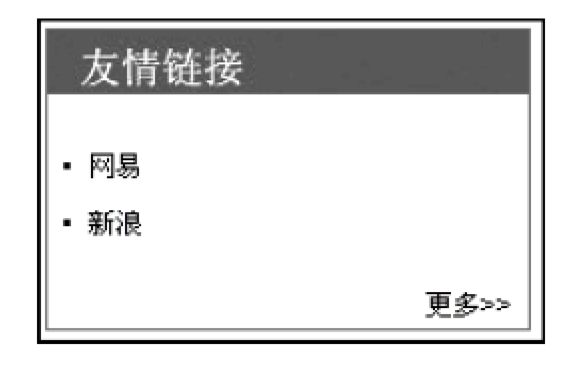


图 5-10　友情链接栏目

要实现这样的效果，其前台代码片段如程序 5-18 和程序 5-19 所示。

程序 5-18　首页中的友情链接栏目（Index.aspx）

```
<table width="210" border="0" align="center" cellpadding="0" cellspacing="0" class="
link01">
    <asp:Label ID="friend" runat="server" ></asp:Label>
  </table>
  <table width="210" border="0" cellspacing="0" cellpadding="0" class="link01">
```

```
    <tr>
      <td width="160" align="right"></td>
      <td width="50"><a href="friendshiplink.aspx?modelname=<%#modelnamefriend %>"
class="link02">更多 &gt;&gt;</a></td>
    </tr>
  </table>
```

程序 5-19　首页中的友情链接栏目后台代码（Index.aspx.cs）

```
protected string modelnamefriend;
protected void Page_Load(object sender, EventArgs e)
 {
       modelnamefriend=this.Page.Server.UrlEncode("> 友情链接");
       StringBuilder sb4=new StringBuilder();
       DataTable FriendTable=new DataTable();
       FriendTable=IndexObject.GetFriendshipLink();
       for (int i=0; i<FriendTable.Rows.Count; i++)
       {
          sb4.Append("<tr>");
          sb4.Append("<td height='25' class='link01'><DIV
             align='left'>").Append(" ").Append(". ").Append (" ").Append("<
             A
             target='_blank' class='link01'
             href='").Append(FriendTable.Rows[i]["LinkUrl"].ToString()).Append("'>").
             Append(FriendTable.Rows[i]["LinkTitle"].ToString()).Append("</A></DIV>
            </td>");
          sb4.Append("</tr> ");
       }
       friend.Text=sb4.ToString();
 }
```

通过调用 IndexObject 的方法 GetFriendshipLink 来获得友情链接的名称及链接地址，按照建立链接的时间排序，得到最新建立的 4 个链接。当用户单击友情链接的名称时，会跳转到相对应的链接页面，当单击“更多”时进入友情链接的栏目页面(显示所有的友情链接列表)。GetFriendshipLink 方法所涉及的 SQL 语句如下所示。

```
select top 4  ID,LinkTitle,LinkUrl
from FriendshipLink
order by AddTime desc
```

首页模块的完整源程序在 Index.aspx 和 Index.aspx.cs 文件中。这里只是对系统简介栏目、培训公告栏目、作业展示栏目和友情链接栏目进行了介绍。对于学习简报栏目、答疑解惑栏目的代码类似于培训公告栏目，思考习题栏目的代码类似于作业展示栏目。

5.4.2 栏目页模块

有了首页,下一步就是栏目页。单击首页各个栏目的“更多”链接,可以进入栏目页。栏目页可以看作是各个栏目的首页,不过它的结构比首页简单,只要包含一个栏目列表就可以了。这里只介绍系统简介栏目和培训公告栏目的栏目页,对于学习简报栏目、答疑解惑栏目、作业展示栏目、思考习题栏目和友情链接栏目与培训公告栏目的栏目页相似,在这里就不作介绍了。

下面首先介绍系统简介栏目的栏目页,如图 5-11 所示。

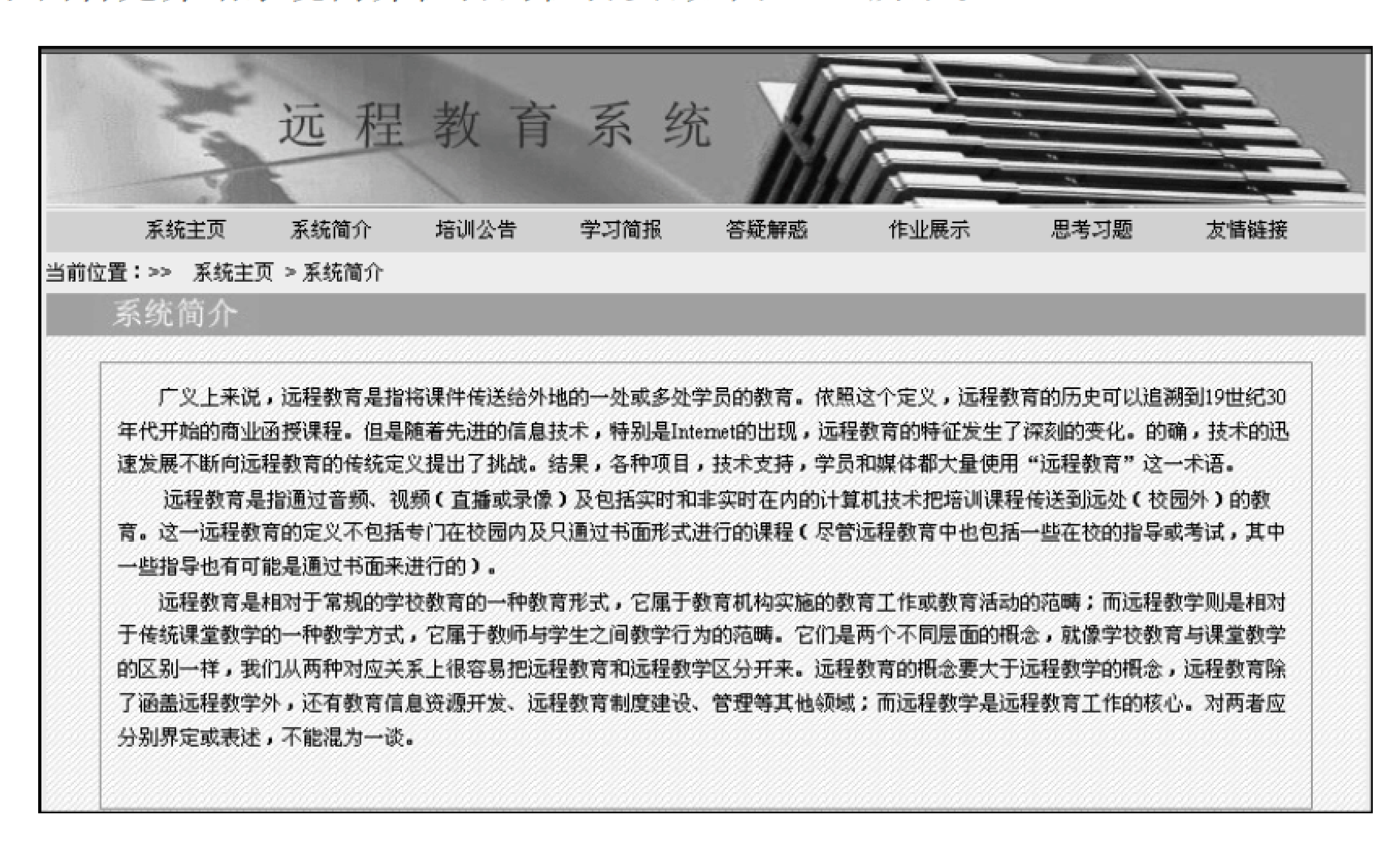


图 5-11　系统简介栏目的栏目页

在此栏目页中,显示了系统简介的完整内容,由此可以对此系统有简单的了解。要实现这样的效果,其前台代码片段如程序 5-20 和程序 5-21 所示。

程序 5-20　栏目页中的系统简介栏目(SystemIntro.aspx)

```
<table width="710" border="0" cellpadding="10" cellspacing="0" class="bd1">
      <tr valign="top">
        <td class="link01" align="left">
            <asp:Label ID="AllSystemIntro" runat="server"></asp:Label>
        </td>
      </tr>
</table>
```

程序 5-21　栏目页中的系统简介栏目后台代码(SystemIntro.aspx.cs)

```
StringBuilder sb1=new StringBuilder();
DataTable IntroTable=new DataTable();
IntroTable=IndexObject.GetSystemIntro();
```

```
string systemintro= IntroTable.Rows[0]["Intro"].ToString();
sb1.Append("        ").Append(sys- temintro.
ToString()).Replace("\r\n", "<br>        ");
AllSystemIntro.Text= sb1.ToString();
```

通过调用 IndexObject 的方法 GetSystemIntro 来获得系统简介的详细内容。sb1.Append("   ")是用来在页面显示时,开头空两格。当把数据库中的内容显示在页面上时,要对格式进行转换,因此要用到 Replace("\r\n", "
        ")对格式进行转换。GetSystemIntro 方法所涉及的 SQL 语句如下所示。

```
select Article.Content as Intro
from [Column] inner join Article on [Column].ID=Article.ColumnID
where [Column].ColumnName='系统简介'
```

系统简介栏目页的完整源程序在 SystemIntro. aspx 和 SystemIntro. aspx. cs 文件中。

接下来实现另一个栏目——培训公告栏目的栏目页,如图 5-12 所示。

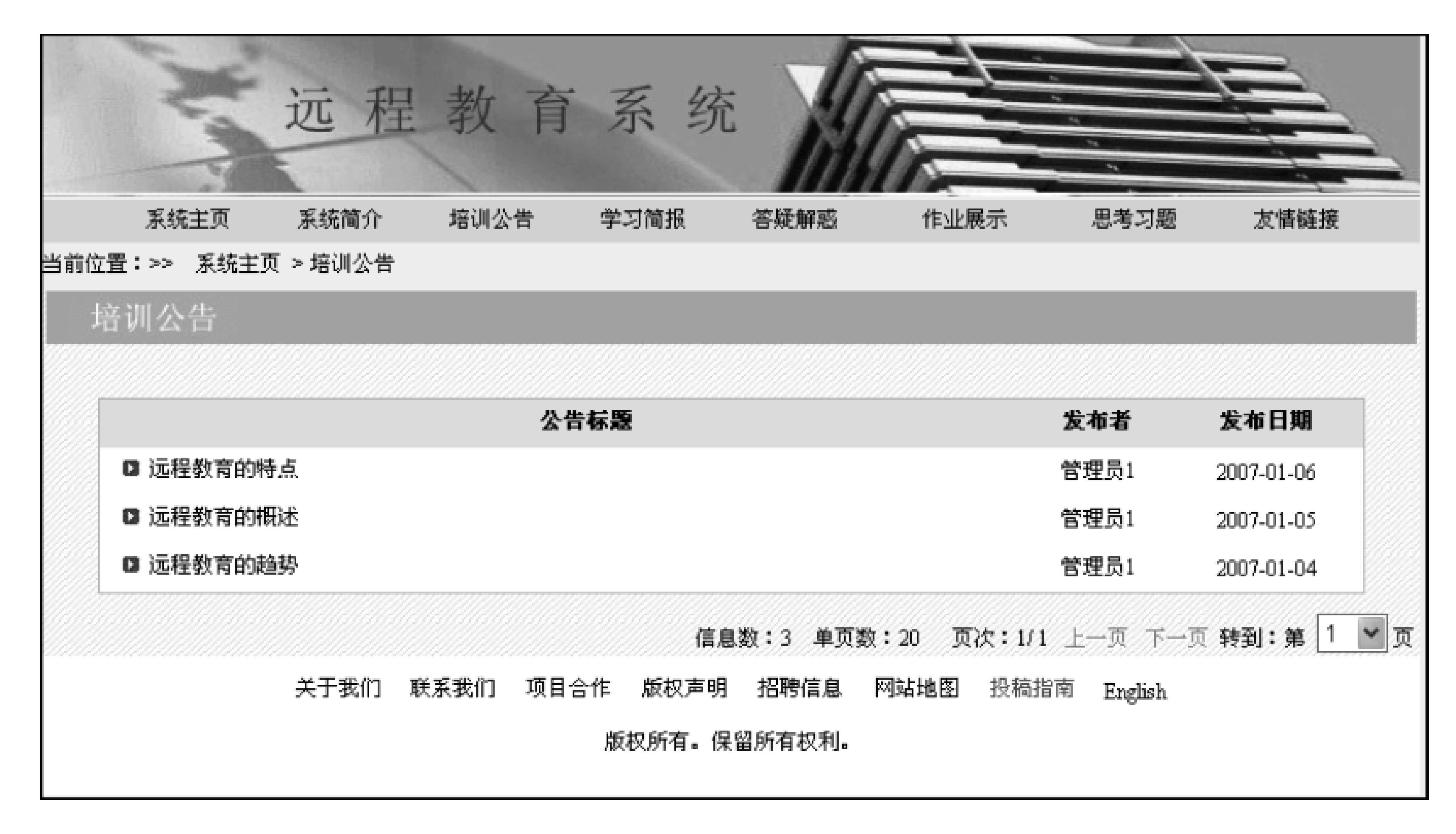


图 5-12　培训公告栏目的栏目页

在此栏目页中,显示了培训公告列表,包括公告标题、发布者和发布日期。在页面的下方设有分页功能。要实现这样的效果,其前台代码片段如程序 5-22、程序 5-23 和程序 5-24 所示。

程序 5-22　栏目页中的培训公告栏目(pxgg. aspx)

```
    …
<asp:DataGrid ID="pxggsum" runat="server"  width="710px" ShowHeader="true"
              AutoGenerateColumns="False" PageSize="20" AllowPaging="True"…>
    <Columns>
     <asp:TemplateColumn >
      <ItemStyle  Width="22px"></ItemStyle>
       < ItemTemplate > < img  src = " image/redarrow. jpg "  width = " 10 "  height = " 9 " >
</ItemTemplate>
        <HeaderStyle BackColor="#EBEBEC" Height="25px" Width="22px" /></asp:Tem-
plateColumn>
    <asp:TemplateColumn HeaderText="公告标题">
      <ItemTemplate>
        <asp:HyperLink  Text='<%#DataBinder.Eval(Container, " DataItem.ArticleTi-
tle")%>'
                    NavigateUrl='<%#
                    myfunc2(DataBinder.Eval(Container.DataItem,"ArticleI2D"))%>' Runat="
                    server"
                    Target="_blank" ID="Hyperlink1" CssClass="link01">
      </asp:HyperLink></ItemTemplate>
        <ItemStyle  CssClass="link01" HorizontalAlign="Left" Height="25px" Width="
450px" ></ItemStyle>
        <HeaderStyle BackColor="#EBEBEC" CssClass="txt" Font-Bold="True" Height="
25px"
                    HorizontalAlign="Center" /></asp:TemplateColumn>
        <asp:BoundColumn DataField="UserName" HeaderText="发布者">
         <ItemStyle HorizontalAlign="Center" CssClass="link01" width="80px" Height="
25px"></ItemStyle>
         <HeaderStyle BackColor="#EBEBEC" CssClass="txt" Font-Bold="True" Height="
25px"
                  HorizontalAlign="Center" Width="80px" /></asp:BoundColumn>
        <asp:BoundColumn DataField="ArticlePublishTime" HeaderText="发布日期">
         <ItemStyle HorizontalAlign="Center" CssClass="link01" width="90px" Height="
25px"></ItemStyle>
         <HeaderStyle BackColor="#EBEBEC" CssClass="txt" Font-Bold="True" Height="
25px"
                  HorizontalAlign="Center" Width="90px" /></asp:BoundColumn>
        </Columns>
      …
  </asp:DataGrid>
      …
<asp:panel id="Panel1" runat="server" Width="100%"  HorizontalAlign="Right">
    <asp:Label id="LabCount" runat="server"></asp:Label>    单
    页数：20
```

```
         
    <asp:Label id="LabCurrent" runat="server"></asp:Label>
    <asp:Label id="LabPageSize" runat="server"></asp:Label>  
    <asp:LinkButton id="LinkButtonPro" runat="server" CommandArgument="prev">上一页
    </asp:LinkButton>  
     < asp: LinkButton id =" LinkButtonNext " runat =" server " CommandArgument =
"next">下一页
    </asp:LinkButton>  转到：第
    <asp:dropdownlist id="Dropdownlist1" runat="server" Width="40px" AutoPostBack="
True">
    </asp:dropdownlist> 页  
</asp:panel>
```

其中 DataGrid 用来绑定数据库显示培训公告列表，panel 中的控件用来实现分页功能。

程序 5-23　栏目页中的培训公告栏目列表后台数据绑定代码（pxgg. aspx. cs）

```
private void BindGrid()
{
    tb=IndexObject.GetAllPxgg();
    pxggsum.DataSource=IndexObject.GetAllPxgg();
    pxggsum.DataBind();
}
```

此段代码用于实现数据绑定，GetAllPxgg()方法用于按照发布时间的顺序显示所有培训公告标题、发布者和发布日期，它所涉及的 SQL 语句如下所示。

```
Select distinct Article.ID as ArticleID, Article.Title as ArticleTitle,
        [User].RealNameasUserName,convert(varchar(10),Article.PublishTime,120) as Arti-
clePublishTime
from [Column] inner join Article on [Column].ID=Article.ColumnID
            inner join [User] on Article.UserID=[User].ID
where [Column].ColumnName='培训公告'
order by ArticlePublishTime desc
```

程序 5-24　栏目页中的培训公告栏目分页后台代码（pxgg. aspx. cs）

```
private void ShowStatsPage()
   { //标签显示页面当前为第几页,共几页信息
       LabCount.Text="信息数："+tb.Rows.Count.ToString();
       LabCurrent.Text="页次："+Convert.ToString((pxggsum.CurrentPageIndex+1))+"/";
        LabPageSize.Text=Convert.ToString(pxggsum.PageCount);
   }
```

```
private void BindDropdownList()
    { //下拉列表显示页数
        int PageCount=pxggsum.PageCount;
        for (int i=1; i<=PageCount; i++)
         {
            Dropdownlist1.Items.Add(i.ToString());
         }
     }

private void SetPageBtnEnable()
        { //设置分页按钮的状态
            if (pxggsum.PageCount==1 || pxggsum.PageCount==0)
            {
                LinkButtonPro.Enabled=false;
                LinkButtonNext.Enabled=false;
                Panel1.Visible=true;
            }
            else
            {
                Panel1.Visible=true;
                if (pxggsum.CurrentPageIndex==0)
                {
                    LinkButtonPro.Enabled=false;
                    LinkButtonNext.Enabled=true;
                }
                else if (pxggsum.CurrentPageIndex==pxggsum.PageCount-1)
                {
                    LinkButtonPro.Enabled=true;
                    LinkButtonNext.Enabled=false;
                }
                else
                {
                    LinkButtonPro.Enabled=true;
                    LinkButtonNext.Enabled=true;
                }
            }
        }
private void PagerButtonClick(object sender, EventArgs e)
      {  //分页按钮的事件处理
         //获得 LinkButton 的参数值
         String arg=((LinkButton)sender).CommandArgument;
         switch (arg)
           {
```

```
        case ("next"):
         if (pxggsum.CurrentPageIndex< (pxggsum.PageCount-1))
          {
            pxggsum.CurrentPageIndex++;
            Dropdownlist1.SelectedIndex=pxggsum.CurrentPageIndex;
           }
           break;
         case ("prev"):
           if (pxggsum.CurrentPageIndex> 0)
            {
               pxggsum.CurrentPageIndex--;
               Dropdownlist1.SelectedIndex=pxggsum.CurrentPageIndex;
             }
             break;
           default:
           break;
       }
        this.BindGrid();
        this.SetPageBtnEnable();
        this.ShowStatsPage();
}
```

以上是对分页部分代码的简单编写，培训公告栏目页的完整源程序在 pxgg. aspx 和 pxgg. aspx. cs 文件中。

5.4.3　文章浏览模块

浏览者最终的目的是阅读文章的内容，在这里只介绍培训公告和作业展示的文章浏览页。对于学习简报和答疑解惑的浏览页与培训公告相似，思考习题的浏览页与作业展示相似。

首先介绍培训公告的浏览页，当浏览者浏览培训公告的文章时会看到培训公告的标题、发布者、发布的时间和正文等信息。所设计的培训公告的浏览页如图 5-13 所示。

从图 5-13 中可以看到，培训公告的文章浏览页主要由三个部分组成。要实现这样的效果，其前台代码片段如程序 5-25 和程序 5-26 所示。

程序 5-25　培训公告的浏览页（pxgg-info. aspx）

```
<table width="700" border="0" align="center" cellpadding="10" cellspacing="0" class="
link01">
  <tr>
   <td height="45" colspan="3" align="left" valign="bottom" bgcolor="#ffffff" class=
"articletitle">
    <asp:Label ID="pxggtitle" runat="server"></asp:Label></td></tr>
  <tr>
```



图 5-13 培训公告的文章浏览页

```
        <td colspan="3" align="center" bgcolor="#ffffff" class="txt">发布者：
         <asp:Label ID="authorname" runat="server"></asp:Label>
                          
                发布时间：<asp:Label ID="publishtime" runat="server"></asp:Label></td
></tr>
    <tr>
        <td colspan="3" align="left" bgcolor="#ffffff" class="txt">
          <p align="left"><asp:Label ID="pxggcontent" runat="server"></asp:Label></p>
</td></tr>
</table>
```

程序 5-26 培训公告的浏览页后台绑定（pxgg-info. aspx. cs）

```
int ArticleID=Convert.ToInt32(Request.QueryString["ArticleID"]);
DataTable PxggContentTable=new DataTable();
PxggContentTable=IndexObject.GetPxggContent(ArticleID);
string PxggTitle=PxggContentTable.Rows[0]["ArticleTitle"].ToString();
pxggtitle.Text=PxggTitle;//培训公告的标题
string AuthorName=PxggContentTable.Rows[0]["UserName"].ToString();
authorname.Text=AuthorName;//培训公告的发布者
string PublishTime=PxggContentTable.Rows[0]["ArticlePublishTime"].ToString();
publishtime.Text=PublishTime;//培训公告的发布时间
```

```
                StringBuilder sb1=new StringBuilder();
                 sb1.Append("        ").Append
(PxggContentTable.Rows[0]["ArticleContent"].ToString()).Replace("\r\n", "<br> 
       ");
                pxggcontent.Text=sb1.ToString();
```

此段代码实现培训公告的数据绑定，GetPxggContent()方法用于通过培训公告的 ID 号得到该公告的相关信息，主要包括公告的标题、公告的发布者、公告发布的时间及公告的内容，其中 Replace("\r\n", "
        ")用来转换数据库中正文的格式，GetPxggContent()方法所涉及的 SQL 语句如下所示。

```
select Article.Title as ArticleTitle,[User].RealName as UserName,convert(varchar(10),
Article.PublishTime,120)
      as ArticlePublishTime,Article.Content as ArticleContent
from  Article inner join [User] on Article.UserID=[User].ID
where Article.ID=@ArticleID
```

培训公告浏览页的完整源程序在 pxgg-info.aspx 和 pxgg-info.aspx.cs 文件中。

接下来介绍作业展示的浏览页，当浏览者浏览作业展示的文章时会看到作业的标题、学生的学号、学生的姓名、作业的要求、学生回答作业的内容、教师的评语及作业的得分等内容。所设计的作业展示的浏览页如图 5-14 所示。

远程教育

学号：13　　姓名：张三

作业要求：

远程教育概述

作业内容：

远程教育是指通过音频、视频（直播或录像）及包括实时和非实时在内的计算机技术把培训课程传送到远处（校园外）的教育。这一远程教育的定义不包括专门在校园内及只通过书面形式进行的课程（尽管远程教育中也包括一些在校的指导或考试，其中一些指导也有可能是通过书面来进行的）。

教师评语：

很好,继续努力!

得分：

98

图 5-14　作业展示的文章浏览页

从图 5-14 中可以看到，作业展示的文章浏览页主要由 7 个部分组成。要实现这样的效果，其前台代码片段如程序 5-27 和程序 5-28 所示。

程序 5-27　作业展示的浏览页（zyzs-info. aspx）

```
<table border="1" style="border-color:Blue; width:600px;" cellpadding="0" cellspacing
="0">
 <tr valign="middle">
  <td align="center" style="height: 50px">
   <asp:Label ID="title" runat="server" Font-Bold="True" Font-Size="Large" ForeColor
="Black" ></asp:Label>
</td></tr>
<tr>
   <td  align="left" style="height: 40px">
    <asp:Label ID="lable1" Text="学号：" runat="server" Font-Bold="True">
</asp:Label>
      <asp:Label ID="id" runat="server" Font-Underline="False"></asp:Label>
                                
       <asp:Label ID="lable2" Text="姓名：" runat="server" Font-Bold="True">
</asp:Label>
      <asp:Label ID="name" runat="server" Font-Underline="False"></asp:Label></td>
</tr>
  <tr>
    <td align="left" style="height: 40px">
        <asp:Label ID="Label1" runat="server" Text="作业要求：" Font-Bold="True"></
asp:Label></td>
   </tr>
   <tr>
  <td align="left" style="height: 120px">
         <asp:TextBox ID="homeworkrequest" runat="server" Height="120px" TextMode="
MultiLine"
              Width="600px" ReadOnly="True"></asp:TextBox></td></tr>
   <tr>
  <td align="left" style="height: 40px">
        <asp:Label ID="Label2" runat="server" Text="作业内容：" Font-Bold="True"></
asp:Label></td>
</tr>
<tr>
   <td align="left" style="height: 300px">
        <asp:TextBox ID="homeworkcontent" runat="server" TextMode="MultiLine" Width=
"600px"
              Height="300px" ReadOnly="True"></asp:TextBox></td></tr>
<tr>
  <td align="left" style="height: 40px">
```

```
        <asp:Label ID="Label3" runat="server" Text="教师评语：" Font-Bold="True"></
asp:Label></td>
  </tr>
  <tr>
   <td align="left" style="height: 100px">
        <asp:TextBox ID="pingyu" runat="server" Height="100px" TextMode="MultiLine"
Width="600px"
              ReadOnly="True"></asp:TextBox></td></tr>
  <tr>
    <td align="left" style="height: 40px">
         <asp:Label ID="Label4" runat="server" Text="得分：" Font-Bold="True"></asp:
Label></td></tr>
  <tr>
   <td align="left" style="height: 50px">
         <asp:TextBox ID="grade" runat="server" Height="50px" TextMode="MultiLine"
Width="600px"
              ReadOnly="True"></asp:TextBox></td></tr>
</table>
```

程序 5-28　作业展示的浏览页后台绑定（zyzs-info.aspx.cs）

```
Session["AnswerID"]=Convert.ToInt32(Request.QueryString["AnswerID"]);
DataTable HomeworkTable=new DataTable();
HomeworkTable = IndexObject. GetHomeworkbyAnswerID ( Convert. ToInt32 ( Session
["AnswerID"]));
string homeworktitle=HomeworkTable.Rows[0]["HomeWorkTitle"].ToString();
title.Text=homeworktitle;                    //作业的标题
string studentid=HomeworkTable.Rows[0]["StudentID"].ToString();
id.Text=studentid;                           //学生的学号
string studentname=HomeworkTable.Rows[0]["StudentName"].ToString();
name.Text=studentname;                       //学生的姓名
string teachercontent=HomeworkTable.Rows[0]["HomeWorkContent"].ToString();
homeworkrequest.Text=teachercontent;         //老师布置作业的要求
string answercontent=HomeworkTable.Rows[0]["AnswerContent"].ToString();
homeworkcontent.Text=answercontent;          //学生回答作业的内容
string teachercomment=HomeworkTable.Rows[0]["TeacherComment"].ToString();
pingyu.Text=teachercomment;                  //老师的评语
string homeworks=HomeworkTable.Rows[0]["HomeworkS"].ToString();
grade.Text=homeworks;                        //分数
```

此段代码实现学生作业的数据绑定，GetHomeworkbyAnswerID()方法用于通过学生回答作业的 ID 号得到该学生的作业答卷，主要包括作业的题目、学生学号、学生的姓名、老师布置作业的要求、学生回答作业的内容、教师的评语及该作业所得的分数，它所涉及的 SQL 语句如下所示。

```
Select HomeWork.Title as HomeWorkTitle, [User].ID as StudentID, [User].RealName as Stu-
dentName, HomeWork.Content as HomeWorkContent, Answer.Content as AnswerContent,Answer.
TeacherComment as
    TeacherComment,Answer.HomeworkS as HomeworkS
from [User] inner join Answer on [User].ID=Answer.StudentID
        inner join HomeWork on HomeWork.ID= Answer.HomeworkID
where Answer.ID=@AnswerID
```

作业展示浏览页的完整源程序在 zyzs-info. aspx 和 zyzs-info. aspx. cs 文件中。

5.4.4 用户登录模块

用户登录模块的功能是控制用户对后台管理程序的访问，图 5-15 是设计的登录页面。

用户登录模块通过输入的用户名和密码来验证用户身份，如果验证失败，拒绝用户访问，将会提示如图 5-16 所示的信息。

如果验证成功，将会进入登录模块的另一个页面，接受用户访问，如图 5-17 所示。



图 5-15 登录页面



图 5-16 登录失败的提示信息

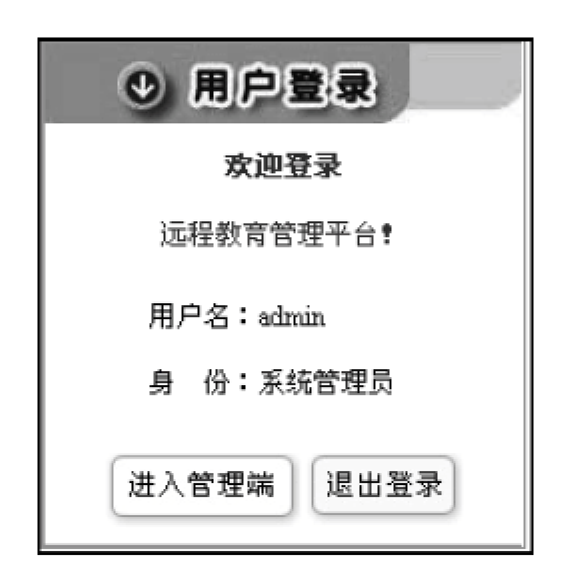


图 5-17 验证成功的页面

当用户单击“进入管理端”按钮时，会进入远程教育的管理平台，如图 5-18 所示；单击“退出登录”按钮则可以退出登录成功的页面，进入远程教育系统的系统主页。



图 5-18 管理平台页面

由图 5-15 登录页面可以看到，它主要由角色选择、用户名、密码、验证码、登录图片、注册图片和忘记密码链接 7 个部分组成。要实现这样的效果，其前台代码片段如程序 5-29 所示。

程序 5-29　登录页 HTML 代码(Login.aspx)

```
<table width="210" border="0" cellpadding="0" cellspacing="0" bgcolor="#FFFFFF" class
="link01">
    <asp:RadioButton ID="RadioButton1" runat="server" Checked="true" CssClass="
link01" GroupName="type" Text="学员" />     
        <asp:RadioButton ID="RadioButton3" runat="server" CssClass="link01"
GroupName="type" Text="教 师 " />
      …
    <asp:RadioButton ID="RadioButton2" runat="server" CssClass="link01"
                         GroupName="type" Text="管 理 员 " />
      <tr>
        <td style="height: 30px"></td>
        <td style="height: 30px">用户名:
          <asp:TextBox ID="TextBox1" runat="server" MaxLength="20"  Width="120px"
TabIndex="1" ></asp:TextBox></td></tr>
      <tr>
        <td style="height: 30px"></td>
        <td style="height: 30px">密　码:
          <asp:TextBox ID="TextBox2" runat="server" MaxLength="20"
                                TextMode="Password" Width="120px" TabIndex="2" >
</asp:TextBox>
        </td></tr>
   <tr>
     <td height="30"></td>
     <td>验证码:
          <asp:TextBox ID="TextBox3" runat="server" MaxLength="8" TabIndex="3" Width
="60px"></asp:TextBox>
          <asp:ImageButton ID="ImageButton1" runat="server" Height="22px" Width="
55px" ImageUrl="~/Login/ImageLog.aspx" /></td></tr>
    <tr>
      <td style="height: 35px"></td>
      <td style="height: 35px"><asp:ImageButton ID="ImageButton2"   runat="server"
width="44" Height="27px"  border="0" align="absmiddle" ImageUrl="../Image/login.gif"
  OnClick="ImageButton2_Click" TabIndex="9"  />  
                    <a href="Register.aspx" target="_blank"><img src="../Image/reg-
ister.gif"
                         width="44" height="27"  border="0" align="absmiddle" /></a
>  
                         <a href="GetPassWord.aspx" class="textblack" target="_blank"
>忘记密码?</a>
```

```
            </td>
        </tr>
    </table>
```

用户单击“登录”按钮触发 Click 事件，如程序 5-30 所示。通过调用 checkloginObject 的方法 CheckStudentUserLogin、CheckTeacherLogin 和 CheckAdminLogin 分别验证学员、教师及管理员的用户名和密码是否合法，如果合法则跳到登录后的页面，如图 5-17 所示。

程序 5-30　Click 事件验证用户信息（Login.aspx.cs）

```
if ((TextBox1.Text=="") || (TextBox2.Text==""))
  {
      Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "sd", "alert('用户名和密码不能为空！')", true);
      return;
  }
if (TextBox3.Text=="")
  {
      Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "sd", "alert('请输入验证码！')", true);
      return;
  }
if (Session["imageString"]==null)
  {
      Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "sd", "alert('请输入验证码！')", true);
      return;
  }
  string imageString=Session["imageString"].ToString();
if (imageString!=TextBox3.Text)
  {
      Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "sd", "alert('输入的验证码不正确！')", true);
      return;
  }
UserData currentUser=null;
if (RadioButton1.Checked) // 验证学员登录
    {
         currentUser= checkloginObject.CheckStudentUserLogin (TextBox1.Text, TextBox2.Text);
          if (currentUser==null)
               {
```

```
                    Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "sd", "
alert('您的用户名或密码错误,\\n或者选择了错误的角色,\\n请查证后重新登录!');",
true);
                    return;
                }
            else
                {
                    Session["Current_User"]=currentUser;
                    Response.Redirect("Loged.aspx");
                 }
        }
if (RadioButton3.Checked) //验证教师登录
        {
              currentUser= checkloginObject.CheckTeacherLogin(TextBox1.Text, TextBox2.
Text);
                  if (currentUser==null)
                    {
                      Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "sd",
"alert('您的用户名或密码错误,\\n或者选择了错误的角色,\\n请查证后重新登录!');",
true);
                       return;
                     }
                  else
                       {
                          Session.Add("Current_User", currentUser);
                          Response.Redirect("Loged.aspx");
                       }
         }
   if (RadioButton2.Checked)    //验证管理员登录
         {
               currentUser= checkloginObject.CheckAdminLogin(TextBox1.Text, TextBox2.
Text);
                   if (currentUser==null)
                    {
                      Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "sd",
"alert('您的用户名或密码错误,\\n或者选择了错误的角色,\\n请查证后重新登录!');",
true);
                        return;
                     }
             else
                     {
                       Session.Add("Current_User", currentUser);
                       Response.Redirect("Loged.aspx");
                      }
      }
```

CheckStudentUserLogin、CheckTeacherLogin 和 CheckAdminLogin 方法所涉及的 SQL 语句如下所示。

```
// CheckStudentUserLogin
select  [User].[ID] as UserID,UserName,RealName,[PassWord],[User].RoleId as RoleID,Ro-
leName
from [User] inner join Role on [User].RoleId=Role.RoleId
where   [User].UserName=Ltrim(@username)
        and [User].password=Ltrim(@password)
        and [User].RoleId=3
```

```
//CheckTeacherLogin
select  [User].[ID] as UserID,UserName,RealName,[PassWord],[User].RoleId as RoleID,Ro-
leName
from [User] inner join Role on [User].RoleId= Role.RoleId
where   [User].UserName=Ltrim(@username)
        and [User].password=Ltrim(@password)
        and [User].RoleId=2
```

```
//CheckAdminLogin
select  [User].[ID] as UserID,UserName,RealName,[PassWord],[User].RoleId as RoleID,Ro-
leName
from [User] inner join Role on [User].RoleId=Role.RoleId
where   [User].UserName=Ltrim(@username)
        and [User].password=Ltrim(@password)
        and [User].RoleId=1
```

由图 5-17 可以看出，该登录以后的页面主要由用户名、身份、进入管理端和退出登录 4 个部分组成。要实现这样的效果，其前台代码片段如程序 5-31 所示。

程序 5-31　登录验证后页面的 HTML 代码(Loged.aspx)

```
<table width="210" border="0" cellpadding="0" cellspacing="0" bgcolor="#FFFFFF" class
="link01">
    <form id="form1" name="form1" method="post" action="">
      <tr valign="bottom">
        <td height="30" colspan="2" class="redtext" align=center><strong>欢迎登录</
strong></td>
      </tr>
      <tr align="center">
        <td height="40" colspan="2" class="redtext">远程教育管理平台!</td>
      </tr>
      <tr>
        <td width="50" height="30"></td>
```

```
            <td width="185" align=left>用户名：<span class="STYLE1"><%# name %>
    </span></td>
      </tr>
      <tr>
         <td height="30"></td>
           <td align=left>身　份：<span class="STYLE1"><%# shenfen %></span></td>
      </tr>
      <tr>
            <td height="15"><img src="../Image/space.gif" width="23" height="3"></td>
             <td><img src="../Image/space.gif" width="23" height="3"></td>
      </tr>
      <tr align="center" valign="top">
            <td height="40" colspan="2">
             <asp:ImageButton  ImageUrl="../Image/entermanage.gif" width="85" height="
    32" border="0" id="IMG1" runat="server" OnClick="IMG1_Click" CssClass="textblack"/>
              <asp:ImageButton ImageUrl="../Image/logout.gif" width="68" height="32" bor-
    der="0" runat="server" id="IMG2" OnClick="IMG2_Click" />
           </td>
         </tr>
       </form>
    </table>
```

如程序 5-32 所示，当用户成功登录以后，会显示用户的用户名和身份两部分。用户成功登录以后，用户的信息会保存在 Session["Current_User"]中。

程序 5-32　成功登录状态(Loged. aspx. cs)

```
//显示用户名
StringBuilder sb1=new StringBuilder();
sb1.Append(((UserData)Session["current_user"]).LoginName);
name=sb1.ToString();
//显示身份
StringBuilder sb2=new StringBuilder();
sb2.Append(((UserData)Session["current_user"]).RoleName);
shenfen=sb2.ToString();
```

如程序 5-33 所示，Loged. aspx. cs 提供了一个退出登录的方法，当用户单击“退出登录”按钮时，程序将清除 Session，并且跳转到开始登录的页面。

程序 5-33　退出登录状态(Loged. aspx. cs)

```
Session["Current_User"]=null;
Response.Redirect("Login.aspx");
```

如程序 5-34 所示，Loged. aspx. cs 还提供了一个进入管理端的方法。当用户单击“进入管理端”按钮时，程序将进行判断，若此时 Session["Current_User"] == null，则 Ses-

sion 失效，提示重新登录；若 Session 不为空，则会跳转到管理平台的页面。

程序 5-34　进入管理端状态（Loged. aspx. cs）

```
if (Session["Current_User"]==null)
    {
             Page. ClientScript. RegisterClientScriptBlock (this. GetType ( ),  " sd",
"alert('用户名和密码失效,请重新登录!');parent.close();", true);
          Response.Write("<script>parent.opener=null;parent.close()</script> ");
    }
  else
    {
          Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "IndexMain",
          "top.location='../IndexMain.aspx';", true);
  }
```

要想在远程教育系统登录，就少不了注册个人信息，它的界面如图 5-19 所示。

图 5-19　注册个人信息

个人信息注册的前台信息填写程序代码如程序 5-35 所示。

程序 5-35　注册个人信息页面的 HTML 代码（Register. aspx）

```
<table style="width:600px;">
  <tr align="left">
    <td style="height: 26px; width: 108px;">
       <asp:Label ID="Label4" runat="server" Text="用户名：">></asp:Label></td>
```

```
            <td style="height: 26px; width: 38px;">
              <asp:TextBox ID="TextBox3" runat="server" MaxLength="18" Width="150px">
</asp:TextBox>
        </td>
        <td style="height: 26px; width: 313px;">
          <asp:Label ID="Label33" runat="server" Text=" * " Width="26px" ForeColor="
Red"></asp:Label>
          <asp:Button ID="Button3" runat="server" BorderStyle="None" Text="验证"
                 OnClick="Button3_Click"  Width="49px" />
           < asp: Label  ID = " Label5"  runat = " server"  Width = " 144px"  ForeColor = " red" >
</asp:Label></td>
    </tr>
    <tr align="left">
        <td style="width: 108px">
          <asp:Label ID="Label6" runat="server" Text="密码: "></asp:Label></td>
        <td style="width: 38px">
          <asp:TextBox ID="TextBox4" runat="server" MaxLength="20" TextMode="Password"
Width="150px"></asp:TextBox></td>
      <td style="width: 313px">
        <asp:Label ID="Label21" runat="server" ForeColor="Red" Text=" * " Width="26px"
></asp:Label>  
         < asp: Label  ID = " Label22"  runat = " server"  Width = " 147px"  ForeColor = " Red" >
</asp:Label></td></tr>
<tr  align="left">
      <td style="width: 108px; height: 27px;" align=left>
        <asp:Label ID="Label8" runat="server" Text="确认密码: " ></asp:Label></td>
      <td style="width: 38px; height: 27px">
        <asp:TextBox ID="TextBox5" runat="server" MaxLength="20" TextMode="Password"
           Width="150px"></asp:TextBox></td>
          <td style="width: 313px; height: 27px;" align=left>
          <asp:Label ID="Label7" runat="server" ForeColor="Red" Text=" * " Width="26px"
></asp:Label>
          <asp:CompareValidator ID="CompareValidator1" runat="server" ErrorMessage="两次
输入密码不一样!"  ControlToCompare="TextBox4" ControlToValidate= "TextBox5"></asp:
CompareValidator>
          <asp:Label ID="Label31" runat="server" ForeColor="red" Text= ""></asp:Label
></td></tr>
    < tr align= "left">
        <td style="width: 108px; height: 27px;" align="left">
           <asp:Label ID="jiaoyanma" runat="server" Text="验证码: "></asp:Label></
td>
        <td style="width: 38px; height: 27px">
```

```
            <asp:TextBox ID="checkyan" runat="server" MaxLength="8" Width= "150px"></
asp:TextBox></td>
        < td style= "width: 313px">
            <asp:Label ID="Label17" runat="server" ForeColor="Red" Text=" * " Width="
26px"></asp:Label>
            <asp:ImageButton ID="ImageButton1" runat="server" Height="22px" Width="55px"
ImageUrl="~/Login/ImageLog.aspx" /> 
            <asp:Label ID="Label20" runat="server" Width="147px" ForeColor="Red"></asp:
Label>< /td></tr>
…
         <asp:ImageButton ID="Button4" runat="server" ImageUrl="../Image/button22.gif"
             OnClick="Button4_Click" Width="63px" /></td>
         <td style="width: 38px; height: 21px">
            < asp:ImageButton ID="Button5" runat="server" ImageUrl="../Image/button09.
gif"
             OnClientClick="return cancleClick()" OnClick="Button5_Click" /></td>
</table>
```

这里需要注意的是，关于验证控件的使用，在前面的章节中已经有所介绍，通过 ErrorMessage 属性指定出错时显示的提示。再来看看它的后台是如何处理这些数据和信息的。因为后台的处理程序代码很长，所以把它分开来讲述，更为全面的代码可以参考程序页\WEB\Login\Register. aspx. cs。

首先看一下图 5-19 上的验证按钮，这里规定，如果数据库中已经存在输入的用户名，则单击验证按钮时会提示“此用户名已被占用”。它是如何实现的呢？程序代码如程序 5-36 所示。

程序 5-36　验证用户名（Register. aspx. cs）

```
if (TextBox3.Text.Trim()=="")
  {
      Label5.Text="用户名不能为空!";
      return;
   }
checklogin userm=new checklogin();
if (userm.CheckUserNameIsAvailable(TextBox3.Text))
  {
      Label5.Text="此用户名可用";
      Session.Add("userNameAvailable", true);
      Button4.Enabled=true;
  }
else
   {
```

```
        Label5.Text="此用户名已被占用";
        Session.Add("userNameAvailable", false);
    }
```

在程序 5-36 中通过调用 checklogin 的方法 CheckUserNameIsAvailable 来验证所输入的用户名是否已经被占用。如果用户名没有被占用，则所输入的用户名可用；反之，所输入的用户名不可用。CheckUserNameIsAvailable 方法所涉及的 SQL 语句如下所示。

```
select * from [User] where username=@userName
```

新增用户的操作是一个典型的数据库 Insert 操作，如程序 5-37 所示。

程序 5-37　新增用户的操作（Register. aspx. cs）

```
protected void Button4_Click(object sender, EventArgs e)
 {
     Button3_Click(sender, e);// 验证用户名是否可用
     if (TextBox4.Text.Trim()=="")
       {
           Label22.Text="密码不能为空!";
           return;
       }
     if (TextBox4.Text!=TextBox5.Text)
       {
           Label31.Text="两次输入密码不一样!";
           return;
       }
     if (TextBox15.Text=="")
       {
         Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "asdf", "a-
lert('邮箱不能为空!');", true);
         return;
       }
     if (checkyan.Text.Trim()=="")
       {
         Label20.Text="验证码不能为空!";
         return;
       }
     bool availableTAG=Convert.ToBoolean(Session["userNameAvailable"]);
     if (availableTAG) // 进行注册
      {
         checklogin userm=new checklogin();
          // 学员注册
          string UserName=TextBox3.Text;
```

```
            string PassWord=TextBox4.Text;
            string RealName=TextBox1.Text;
            …
            try
              {
                 if (TextBox10.Text=="")
                  { }
              }
       catch
        {
           Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "asdf", "alert('
电话号码应为数字!');", true);
            return;
        }
       try
         {
            if (TextBox9.Text=="")
              { }
             else
               Convert.ToInt64(TextBox9.Text);
          }
         catch
          {
              Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "asdf","a-
lert('手机号码应为数字!');", true);
               return;
            }
         try
           {
              if (TextBox11.Text=="")
               { }
            else
               Convert.ToInt64(TextBox11.Text);
            }
          catch
           {
              Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "asdf","a-
lert('邮政编码应为数字!');", true);
               return;
            }
       if (Session["imageString"]==null)
          {
             Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "sd", "alert
('请输入验证码!')", true);
```

```
                return;
            }
          string imageString=Session["imageString"].ToString();
        if (imageString!=checkyan.Text)
          {
                Page. ClientScript. RegisterClientScriptBlock (this. GetType ( ),  " sd",
"alert('输入的验证码不正确!')", true);
              return;
            }
         int TAG1=userm.InsertUser(UserName, PassWord, RealName, IDCardNumber, Gender,
City, Workplace, TechPost, Duty, Address, Telephone, PostNumber, Cellphone, E-mail, QQ,
MSN);
        if (TAG1==1)
        {
            Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "asdf", "alert('
注册成功!');window.close();", true);
             Label22.Text="";
             Label31.Text="";
             Label20.Text="";
             Button3.Enabled=false;
             Button4.Enabled=false;
             Button5.Enabled=false;
          }
        else
          {
            Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "asdf",
            "alert('注册失败!');window.close();", true);
            Label22.Text="";
            Label31.Text="";
            Label20.Text="";
            Button5.Enabled=false;
          }
      }
}
```

在程序 5-37 中通过调用 checklogin 的方法 InsertUser 向数据库中插入一条记录，添加用户信息。InsertUser 方法所涉及的 SQL 语句如下所示。

```
insert into
[User](UserName,PassWord,RoleId,RealName,IDCardNumber,Gender,City,Workplace,TechPost,
Duty,Address,Telephone,PostNumber,Cellphone,E-mail,QQ,MSN)
values(@UserName,@PassWord,3,@RealName,@IDCardNumber,@Gender,@City,@Workplace,@
TechPost,@Duty,@Address,@Telephone,@PostNumber,@Cellphone,@E-mail,@QQ,@MSN)
```

通过上面的介绍，已经对登录和注册有了简单了解。那么，当用户注册以后，在系统

主页登录时忘记了密码怎么办呢？这就会涉及到怎样找回密码的问题。下面是设计的找回密码的页面，如图 5-20 所示。

图 5-20　找回密码

从图 5-20 中可以看到，该页面由输入真实姓名、身份证号码、确定和取消 4 个部分组成。找回密码的前台程序代码如程序 5-38 所示。

程序 5-38　找回密码页面的 HTML 代码（GetPassWord.aspx）

```
<table style="margin-left:60px;" >
              < tr><td><asp:Label ID="Label1" runat="server" Text="请输入您的真实姓
名和身份证号码"></asp:Label> </td></tr>
</table>
<table width="400"style="margin-left:60px;" >
 <tr>
  <td style="width: 130px">
      <asp:Label ID="Label2" runat="server" Text="真实姓名："></asp:Label></td><td
colspan="2">
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><asp:RequiredFieldValida-
tor
          ID="RequiredFieldValidator1" runat="server" ErrorMessage="姓名不能为空!"
          ControlToValidate="Textbox1"></asp:RequiredFieldValidator></td></tr>
<tr>
  <td style="width: 130px">
   <asp:Label ID="Label3" runat="server" Text="身份证号码："></asp:Label></td><td
colspan="2">
   <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox> <asp:RequiredFieldVali-
dator
          ID= "RequiredFieldValidator2" runat="server" ErrorMessage="身份证号码不能
为空!"
          ControlToValidate="Textbox2"></asp:RequiredFieldValidator></td></tr>
<tr>
  <td style="width:130px; height: 13px;">
      < asp: ImageButton ID="Button1" OnClick="Button1_Click" ImageUrl="../Image/
btnok.gif"
```

```
        runat="server" /></td>
    <td colspan="2" style="height: 13px">
    <asp:ImageButton ID="ImageButton2" OnClientClick="return cancel()" ImageUrl="../Im-
age/backnew.gif"
        runat="server" /></td></tr>
<tr>
    <td style="width:130px; height:50px">
            <asp:Label ID="mima" runat="server" Text="您的密码是：" Visible=
"False" Height="40px"
            Font-Bold="True" Font-Size="10pt" ForeColor="Red" Width="136px">
</asp:Label></td>
    <td align="left"><asp:Label ID="pass" runat="server" Visible="False" Font- Bold="
True" ForeColor="Red" Font-Size="10pt" Height="40px" Width="248px"> </asp:Label></td
></tr>
<tr>
    <td>
      <asp:Label ID="Label6" runat="server" Visible="False" Font-Bold="True" Font-Size
="Small" ForeColor="ActiveCaption" Height="30px" Width="144px"> </asp:Label></td></
tr>
</table>
```

它是怎样通过输入的真实姓名和身份证号码找回密码的呢？程序代码如程序 5-39 所示。

程序 5-39　找回密码（GetPassWord.aspx.cs）

```
protected void Button1_Click(object sender, EventArgs e)
  {
            int typeTag;
            checklogin userm=new checklogin();
            typeTag=userm.GetUserName(TextBox1.Text, TextBox2.Text);
            Session.Add("typeTag", typeTag);
               if (typeTag==2)
                    {
                        Label6.ForeColor=Color.Red;
                        Label6.Text="您还没有注册!";
                        Label6.Visible=true;
                        return;
                      }
               else if (typeTag==-1)
                    {
                      Label6.ForeColor=Color.Red;
                           Page.ClientScript.RegisterClientScriptBlock(this.GetType(),
"", "alert('您输入的姓名和身份证号码错误!')", true);
```

```
                    Label6.Visible=false;
                    return;
                }
                else
                {
                    TextBox1.Enabled=false;
                    TextBox2.Enabled=false;
                    Button1.Enabled=false;
                    mima.Visible=true;
                    pass.Visible=true;
                    //找回密码
                    string realname=TextBox1.Text;
                    string idnumber=TextBox2.Text;
                    DataTable PassWordTable=new DataTable();
            PassWordTable=
            IndexObject.GetPassWordbyRealNameAndIDCardNumber(realname,idnumber);
                    string password=PassWordTable.Rows[0]["PassWord"].ToString();
                    pass.Text=password;//密码
                    ImageButton2.Enabled=false;
                }
        }
```

在程序 5-39 中通过调用 checklogin 的方法 GetUserName 来验证该用户是否已经注册,若已经注册,则可以找回密码;反之,不能找回密码。通过调用 IndexObject 的方法 GetPassWordbyRealNameAndIDCardNumber,由输入的真实姓名和身份证号码找回密码。GetUserName 方法所涉及的 SQL 语句前面已经介绍过。

PassWordbyRealNameAndIDCardNumber 方法所涉及的 SQL 语句如下所示。

```
select  PassWord
from [User]
where RealName=@RealName
      and
IDCardNumber=@IDCardNumber
```

登录模块的完整源程序在\WEB\Login 文件夹中。

5.4.5 导航菜单模块

图 5-21 所示为系统管理员登录后看到的后台管理页面 IndexMain.aspx。该页面由三部分组成,分别为头部(页面上方)、主体(页面中间)和尾部(页面下方)。主体部分又分为两部分:左边部分和右边部分。导航菜单模块就是左边部分。

学员和教师登录后,进入的也是这个页面,区别在于导航菜单中的内容不同,因为导航菜单是根据角色动态加载的。下面来具体介绍导航菜单实现的原理。

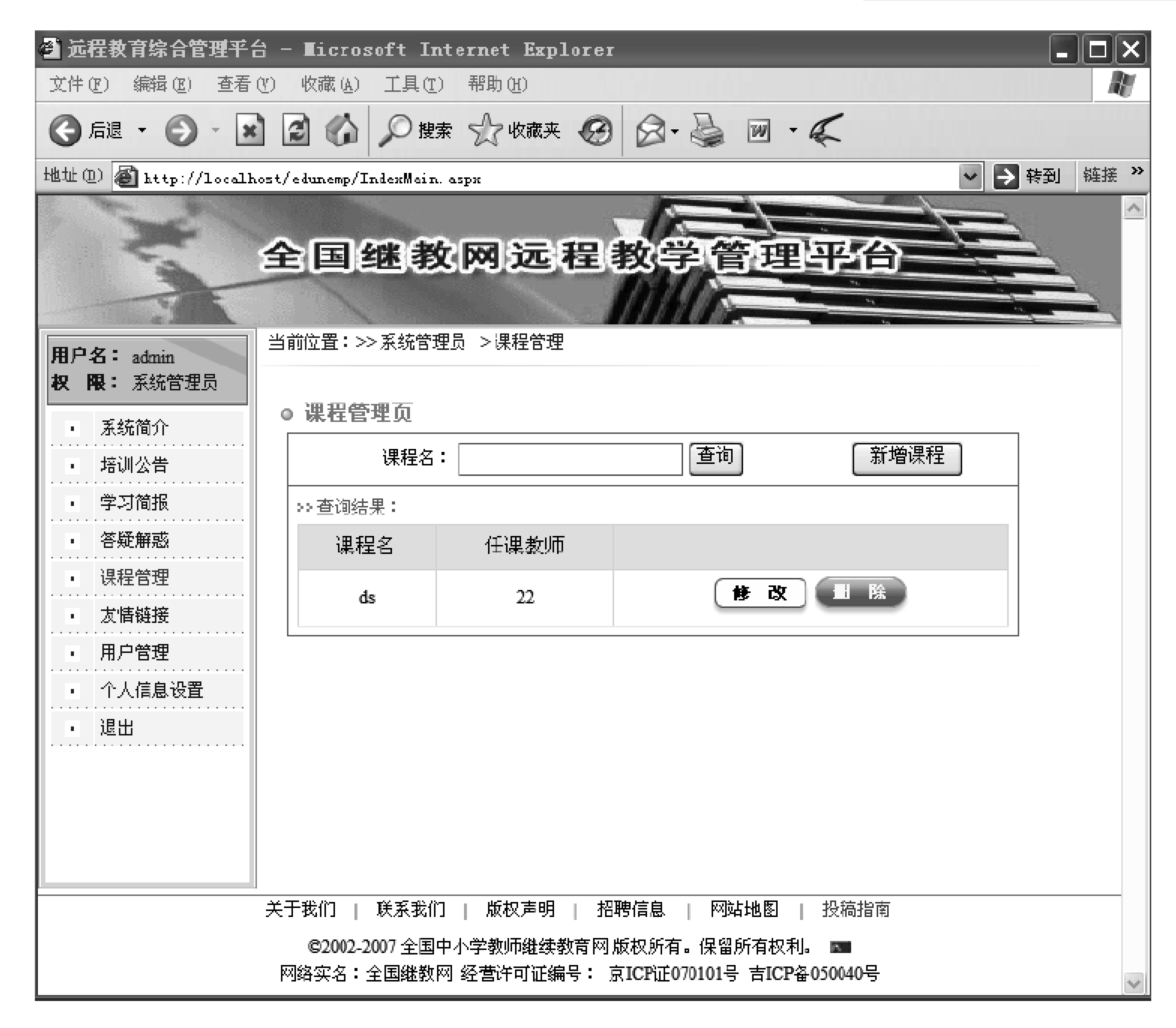


图 5-21　后台管理页面(IndexMain.aspx)

如果把菜单中的每一项称为一个模块，那么每个角色所对应的模块是不一样的。角色与模块的关系是通过一个 XML 来配置的。下面先介绍该 XML，如程序 5-40 所示。

程序 5-40　InitList.xml 的部分内容

```
<?xml version="1.0" encoding="utf-8" ?>
<AccessPage>
  <InitPage>
    <Role>1</Role>
    <ModuleName>系统简介</ModuleName>
    <ModuleEntrance>BulletinManage/BulletinList.aspx?ColumnID=1</ModuleEntrance>
  </InitPage>
  …
  <InitPage>
    <Role>1</Role>
    <ModuleName>课程管理</ModuleName>
    <ModuleEntrance>CourseManage/Default.aspx</ModuleEntrance>
  </InitPage>
  <InitPage>
    <Role>1</Role>
    <ModuleName>友情链接</ModuleName>
```

```
        <ModuleEntrance>LinkManage/Default.aspx</ModuleEntrance>
    </InitPage>
    <InitPage>
        <Role>1</Role>
        <ModuleName>用户管理</ModuleName>
        <ModuleEntrance>UserManage/Default.aspx</ModuleEntrance>
    </InitPage>
    <InitPage>
        <Role>1</Role>
        <ModuleName>个人信息设置</ModuleName>
        <ModuleEntrance>IndividualInformation/perinformation.aspx</ModuleEntrance>
    </InitPage>
    <InitPage>
        <Role>1</Role>
        <ModuleName>退出</ModuleName>
        <ModuleEntrance>LoginOut.aspx</ModuleEntrance>
    </InitPage>
    <InitPage>
        <Role>2</Role>
        <ModuleName>作业展示</ModuleName>
        <ModuleEntrance>Teacher/TeacherHomework.aspx</ModuleEntrance>
    </InitPage>
    <InitPage>
        <Role>2</Role>
        <ModuleName>思考习题</ModuleName>
        <ModuleEntrance>CourseStudy/Default.aspx</ModuleEntrance>
    </InitPage>
    …
    <InitPage>
        <Role>3</Role>
        <ModuleName>作业展示</ModuleName>
        <ModuleEntrance>Student/Default.aspx</ModuleEntrance>
    </InitPage>
    <InitPage>
        <Role>3</Role>
        <ModuleName>思考习题</ModuleName>
        <ModuleEntrance>CourseStudy/Default.aspx</ModuleEntrance>
    </InitPage>
    …
</AccessPage>
```

由于篇幅有限，上面是 XML 文件的部分内容。该 XML 文件名为 InitList.xml，根节点为 AccessPage，根节点包含很多子节点 InitPage，其中每个 InitPage 子节点对应一个菜

单中的模块。下面详细介绍 InitPage 子节点，其形式如下。

```
<InitPage>
    <Role>1</Role>
    <ModuleName>系统简介</ModuleName>
    <ModuleEntrance>BulletinManage/BulletinList.aspx?ColumnID=1</ModuleEntrance>
  </InitPage>
```

每个子节点 InitPage 下面都包含三个子节点 Role、ModuleName 和 ModuleEntrance。Role 是对应数据库中的角色编号；ModuleName 代表模块的名称；ModuleEntrance 为模块的链接地址，即模块所对应页面的相对地址。

在该系统中，管理员有 9 个模块，分别为系统简介、培训公告、学习简报、答疑解惑、课程管理、友情链接、用户管理、个人信息设置和退出。所以在 InitList. xml 中应有 9 个 InitPage 节点。在这 9 个模块中，Role 中的文本都应为管理员的角色编号 1，ModuleName 中的文本对应这 9 个模块的名称，ModuleEntrance 为 9 个模块的链接地址。例如，系统简介的 ModuleEntrance 为 BulletinManage/BulletinList. aspx? ColumnID=1，因为系统简介在 BulletinManage 文件夹的 BulletinList. aspx 中。由于系统简介、培训公告等都共用该页面，所以加上查询字符串？ColumnID=1 以区分不同模块。

学员和教师类似。如图 5-22 和图 5-23 所示。



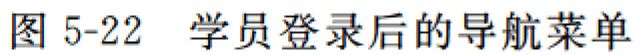

图 5-22　学员登录后的导航菜单



图 5-23　教师登录后的导航菜单

程序 5-41 所示为导航菜单的前台 HTML 代码。

程序 5-41　导航菜单的 HTML 代码(IndexMain. aspx)

```
<asp:DataList ID="DlMenusTop" Width="100%" CellPadding="0" CellSpacing="0" runat="
server"BorderWidth="0" >
<ItemTemplate>
    <table cellspacing="1" cellpadding="4" width="100%" align="center" border="0">
        <tr>
          <td height=24 background="Image/tit027.gif" width="15%">
                <img src="Image/tit015.gif" width="11" height="11" hspace="5">
          </td>
```

```
            <td height=24 background="Image/tit027.gif">
                <span>
<asp:HyperLink ID="linkItem" runat="server"  CssClass="link04"
Text=<%#DataBinder.Eval(Container.DataItem,"ModuleName").ToString()%>
NavigateUrl='<%#DataBinder.Eval(Container.DataItem, "ModuleEntrance")%>'
Target=frmLevelTree >
</asp:HyperLink>
              </span>
             </td>
          </tr>
      </table>
    </ItemTemplate>
</asp:DataList>
```

需要注意的是，服务器控件 HyperLink 的 Text 和 NavigateUrl 都是后台绑定，稍后会讲到。Target 属性的值为 frmLevelTree，要理解这里需要清楚主体部分右边的作用。右边是一个 iframe，用来承载左边导航的页面。其前台 HTML 代码如程序 5-42 所示。

程序 5-42　主体右边部分的 HTML 代码（IndexMain. aspx）

```
<table border=0 width=100% height=100% align=center id=tbFrame>
    <tr height=100%>
      <td align=center valign=top>
          <iframe id=frmLevelTree frameBorder="0" runat=server
             src="" scrolling=auto width="100%" height=100% name="frmLevelTree">
          </iframe>
      </td>
    </tr>
</table>
```

Iframe 的 ID 为 frmLevelTree，正好是前面讲到的服务器控件 HyperLink 的 Target 的值，表明导航菜单链接的页面都是在 Iframe 显示，也就是页面主体部分的右边。

接下来介绍实现导航菜单的后台核心代码，如程序 5-43 所示。

程序 5-43　导航菜单实现的后台代码(IndexMain. aspx. cs)

```
DataSet ds=new DataSet();
ds.ReadXml(HttpContext.Current.Server.MapPath(@"xml/InitPage.xml"));
DataView dv=ds.Tables[0].DefaultView;
dv.RowFilter="Role="+roleId.Trim();
DlMenusTop.DataSource=dv;
DlMenusTop.DataBind();
```

首先把指定位置的 XML 文件中模块信息读取到数据集中，再根据视图的过滤属性，过滤掉不是目前登录角色所应有的模块信息，并以此视图为数据列表 DlMenusTop 的数

据源进行绑定，从而达到可以根据不同登录用户的角色显示不同的菜单信息。

到这里，导航菜单已经介绍完毕，所有代码都在系统根目录下 IndexMain. aspx、IndexMain. aspx. cs 中。

5.4.6　文章管理功能模块

1. 功能描述

文章管理功能模块分为系统简介、培训公告、学习简报和答疑解惑 4 个版块。对所有版块文章进行管理，有查询、发布、修改和删除 4 种操作，不同的用户角色享有不同的权限。只有管理员能对文章进行查询、发布、修改和删除等操作，其他学员都只能进行浏览。系统简介、培训公告、答疑解惑和学习简报等管理模块在具体实现上基本相同，在此仅以培训公告管理版块为例，就不再一一具体介绍。

2. 页面实现

显示文章列表的管理页面的实现代码参见 5.3 节中分层结构部分的详细介绍，相关文件为 BulletinList. aspx 和 BulletinList. aspx. cs。

显示文章列表的管理页面如图 5-24 所示。

当前位置：>>　系统主页

（本区域内显示在本栏目下查询到的的文章列表）

标题	发布者	发布时间	修 改	删 除
远程教育的特点	管理员1	2007-1-6 0:00:00	修 改	删除
远程教育的趋势	管理员1	2007-1-4 0:00:00	修 改	删除
远程教育的概述	管理员1	2007-1-5 0:00:00	修 改	删除

发表新文章

图 5-24　显示文章列表的管理页面

增加文章页面(ArticleAdd. aspx)的 Html 实现代码如程序 5-44 所示。

程序 5-44　ArticleAdd. aspx

```
<table width="772" height="20" border="0" align="center" cellpadding="0" cellspacing=
"0">
            <tr>
                <td style="width: 55px; height: 3px">标题: </td>
                <td style="width: 550px; height: 3px" align="left">
                  <asp:TextBox ID="TextBoxTitle" runat="server" Width="550px"></asp:
TextBox>
                  <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="
server" ControlToValidate="TextBoxTitle"   ErrorMessage="文章标题不能为空" Font-Size
="Small"
```

```
Width="136px"></asp:RequiredFieldValidator></td>
                </tr>
               <tr>
                   <td style="width: 60px; height: 28px">发布人:</td>
                    <td style="width: 232px; height: 28px" align="left">
                    <asp:Label ID="Label1"  runat="server" Height="23px"></asp:Label>
</td>
               </tr>
               <tr>
                   <td style="width: 60px; height: 4px">内容:</td>
                     <td colspan="2" style="height: 302px; width: 712px;">
                          < asp:TextBox ID="TB_Content" runat="server" Height= "295px"
TextMode="MultiLine" Width="708px"></asp:TextBox>
               </tr>
               <tr>
                   <td align="center" colspan="2">
                        <asp:ImageButton ID="ButtonOK" runat="server" Height="23px" Width
="80px" OnClick="ButtonOK_Click" ImageUrl="../Image/SubmitArticle.jpg" />
               </tr>
          </table>
```

(ArticleAdd.cs 文件参见上一节三层结构中的相关文件)

增加培训公告文章的页面效果如图 5-25 所示。

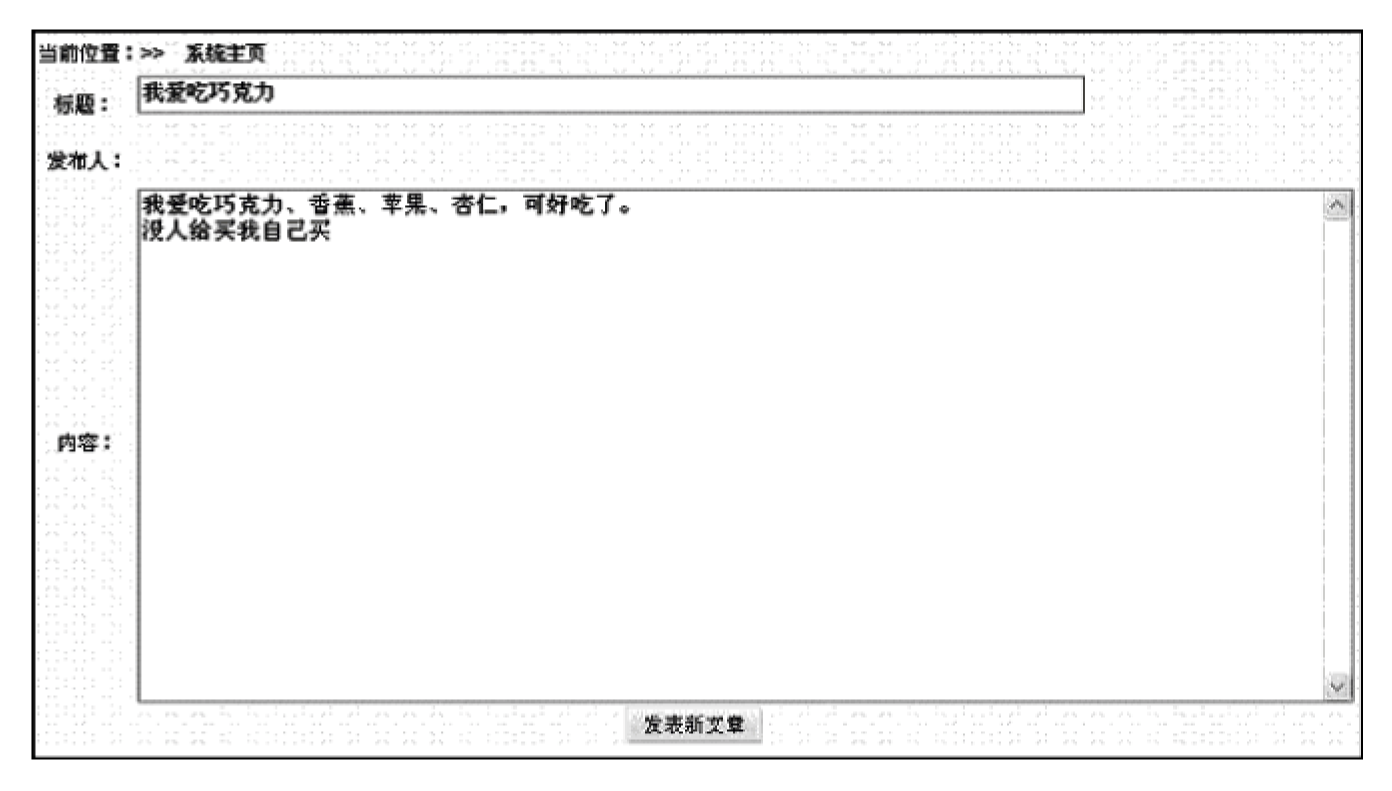


图 5-25 发表新文章页面

修改文章的功能是指当有管理权限的用户单击相应文章的修改按钮时跳转到的页面，页面的控件中显示文章的原有内容。当用户编辑完后单击“确定”按钮将修改数据库中的文章信息。

修改文章的页面(ArticleEdit.aspx)Html 代码如程序 5-45 所示。

程序 5-45 ArticleEdit.aspx

```
        <table width="772" height="20" border="0" align="center" cellpadding="0"
cellspacing="0">
            <tr>
                <td style="width: 55px; height: 3px">标题：</td>
                <td style="width: 650px; height: 3px" align="left">
                    <asp:TextBox ID="TextBoxTitle" runat="server" Width="650px"></
asp:TextBox>
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="
server" ControlToValidate="TextBoxTitle"  ErrorMessage="文章标题不能为空" Font-Size
="Small"Width="136px"></asp:RequiredFieldValidator></td>
            </tr>
            <tr>
                <td style="width: 60px; height: 28px">发布人：</td>
                 <td style="width: 232px; height: 28px" align="left">
                    <asp:Label ID="Label1" runat="server" Height="23px"></asp:Label
></td>
            </tr>
            <tr>
                <td style="width: 60px; height: 4px">内容：</td>
                 <td colspan="2" style="height: 302px; width: 712px;">
                    <asp:TextBox ID="TB_Content" runat="server" Height="295px" Text-
Mode="MultiLine"
                                Width="708px"></asp:TextBox></td>
            </tr>
            <tr>
                <td align="center" colspan="2">
                    <asp:ImageButton ID="ButtonOK" runat="server" Width="56px"Height
="25px" OnClick="ButtonOK_Click"
                          ImageUrl="../Image/BtnOk.gif" CssClass="link05"></asp:Im-
ageButton>
            </tr>
        </table>
```

（ArticleEdit.cs 文件参见上一节三层结构中的相关文件）

修改文章的页面效果如图 5-26 所示。

5.4.7　用户管理模块

用户管理模块是管理员才具有权限操作的模块。具有的操作权限为查询、删除和增加。该模块主要由 UserManage 文件夹下的 Default.aspx 和 AddUser.aspx 两个页面实现。Default.aspx 如图 5-27 右边所示。

其中用户列表由 GridView 控件实现，其前台的 HTML 页面的代码如程序 5-46 所示。

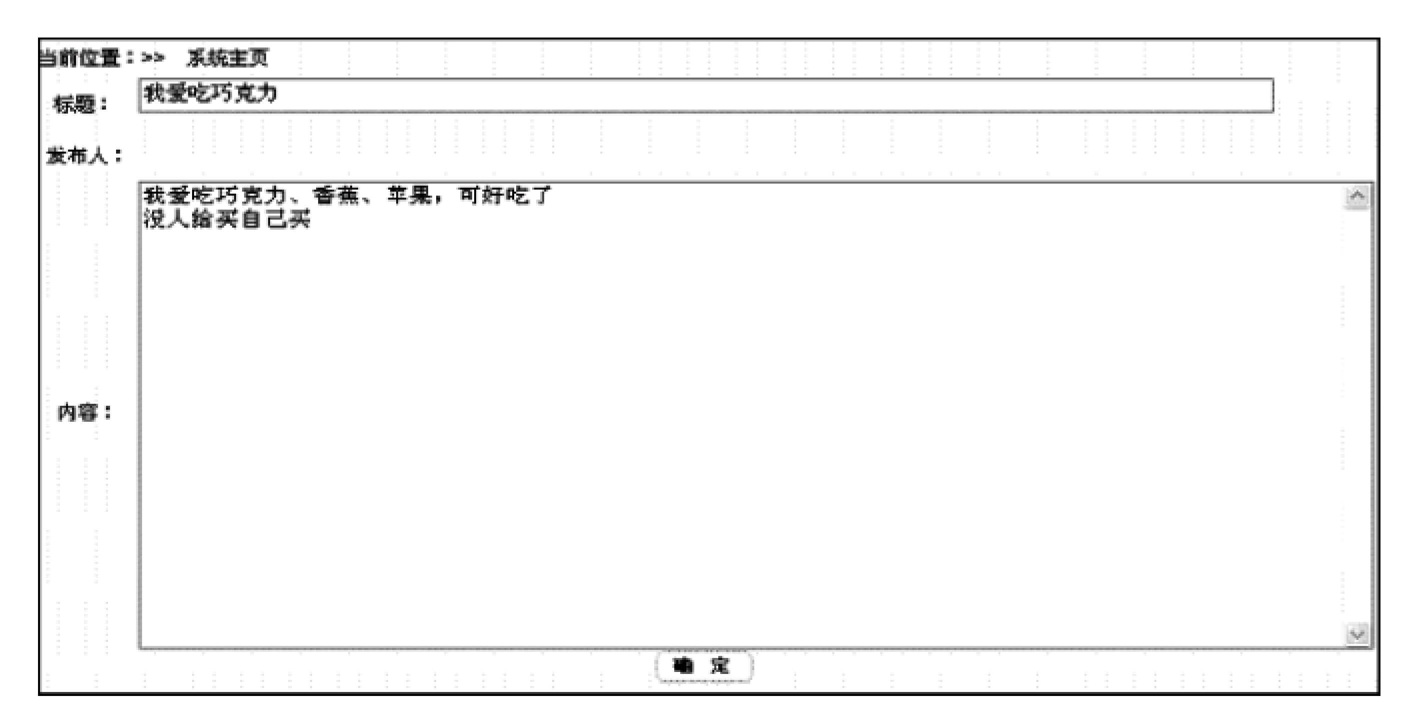


图 5-26 修改文章页面



图 5-27 用户管理模块首页(Default. aspx)

程序 5-46 GridView 控件的前台 HTML 代码(Default. aspx)

```
< asp:GridView ID="GV_UserList" runat="server" AllowPaging="True"
     AutoGenerateColumns="False"  ageSize="20"
     OnRowCommand="GV_UserList_RowCommand">
  <Columns>
    <asp:BoundField DataField="UserName" HeaderText="用户名" HeaderStyle-
Font-Bold=false/>
    <asp:BoundField DataField="PassWord" HeaderText="密码"  HeaderStyle-
Font-Bold=false/>
    <asp:BoundField DataField="RoleName" HeaderText="角色名称" HeaderStyle
-Font-Bold=false/>
    <asp:TemplateField  HeaderStyle-Font-Bold=false>
       <ItemTemplate>
```

```
                    <asp:ImageButton ID="LB_DeleteUser"  runat="server" class="link02"
                     CommandName="DeleteUser" OnClientClick="return confirm('确定删除?
')"
                    CommandArgument='<%#Eval("ID") %>'  ImageUrl="~/Image/BtnDelete.
gif" />
                    </ItemTemplate>
                </asp:TemplateField>
              </Columns>
           </asp:GridView>
```

由代码可知，用户列表是由服务器控件 GridView 在后台动态绑定的，具体实现如程序 5-47 所示。

程序 5-47　用户列表绑定的后台代码(Default. aspx. cs)

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        GV_DataBind(GetUserList(""));
    }
}
private DataTable GetUserList(string UserName)
{
    object[] Params=new object[] { UserName };
    DataTable dt=UserManageObject.GetUsers(Params);
    return dt;
}
private void GV_DataBind(DataTable dt)
{
    this.GV_UserList.DataSource=dt.DefaultView;
    this.GV_UserList.DataBind();
    }
```

当首次加载页面时，调用 GV_DataBind 方法，该方法的参数是另一个方法 GetUserList 的返回值。GetUserList 方法主要由 UserManageObject 类的 GetUsers 方法实现。

UserManageObject 是逻辑层的用户管理类，包装了与用户管理相关的方法。GetUsers 方法是获取用户列表，参数为用户名。当用户名为空时，获取全部用户列表；当用户名不为空，模糊查询，获取满足条件的用户列表。返回值为一个 DataTable 型的列表。

GetUserList 方法中定义一个 DataTable 型的变量 dt 接收 GetUsers 方法的返回值，并将 dt 返回。

GV_DataBind 方法以 dt 为参数，并将 dt 作为服务器控件 GV_UserList 的数据源，进行绑定，从而显示全部用户列表。

其中,GetUsers 方法设计的 SQL 语句如下所示。

当参数为空时,SQL 语句如下。

```
Select U.[ID],U.UserName,U.[PassWord],R.RoleName
                    FROM [User] U inner join Role  R ON
                    U.RoleID=R.RoleID
```

当参数不为空时,SQL 语句如下。

```
Select U.[ID],U.UserName,U.[PassWord],R.RoleName
                    FROM [User] U inner join Role  R ON
                    U.RoleID=R.RoleID
                    WHERE U.UserName LIKE  '%'+@UserName+'%'
```

用户的删除是由 GridView 控件的 RowCommand 事件完成,CommandName＝"DeleteUser"指定了命令名称。CommandArgument＝'<%#Eval("ID")%>'表明命令的参数是后台绑定的用户编号。其具体实现如程序 5-48 所示。

程序 5-48　删除用户的实现(Default.aspx.cs)

```
protected void GV_UserList_RowCommand(object sender, GridViewCommandEventArgs e)
        {
            string CommandName=e.CommandName;
            string UserID=e.CommandArgument.ToString();
            switch (CommandName)
            {
                case "DeleteUser":
                    {
                        Delete(UserID);
                        break;
                    }
                default:
                    {
                        break;
                    }
            }
            string UserName=this.txt_UserName.Text.Trim();
            GV_DataBind(GetUserList(UserName));
        }
        protected void Delete(string UserID)
        {
            object[] Parmas=new object[]{UserID};
            UserManageObject.DeleteUser(Parmas);
        }
```

当单击“删除”按钮时，触发 GridView 控件的 RowCommand 事件 GV_UserList_RowCommand。根据用户命令 DeleteUser，会调用 Delete 方法。Delete 方法主要有 UserManageObject 类的 DeleteUser 方法。DeleteUser 方法为删除指定用户，参数为用户编号。Delete 方法执行完后，会重新绑定，显示删除后的用户列表。DeleteUser 方法涉及的 SQL 语句如下。

```
DELETE  FROM  [User] WHERE [ID]=@UserID
```

用户的查询和新增的前台 HTML 代码如程序 5-49 所示。

程序 5-49　查询和新增的 HTML 代码(Default.aspx)

```
<table width="100%">
   <tr>
     <td align="right" width="15%">用户名:</td>
     <td align="left" width="15%">
        <asp:TextBox ID="txt_UserName" runat="server"></asp:TextBox></td>
     <td align="left" width="15%">
        <asp:Button ID="select" runat="server" Text="查询" OnClick="select_Click"  /
></td>
     <td align="left" width="15%">
        <asp:Button ID="Add" runat="server" Text="新增用户" OnClick="Add_Click"
   />
     </td>
   </tr>
</table>
```

单击“查询”按钮和“新增”按钮，触发的事件分别为 select_Click、Add_Click。后台代码如程序 5-50 所示。

程序 5-50　新增和查询时间后台代码(Default.aspx.cs)

```
protected void Add_Click(object sender, EventArgs e)
{
    string url="AddUser.aspx";
    Response.Redirect(url);
}
protected void select_Click(object sender, EventArgs e)
{
    string UserName=this.txt_UserName.Text.Trim();
    GV_DataBind(GetUserList(UserName));
}
```

查询用户跟前面初始化用户列表的代码类似，不同的是：查询用户时，是根据用户名模糊查询，参数不为空，用到的 SQL 语句是前面介绍的参数不为空的 SQL 语句，所以得到的是满足条件的用户，而并非全部用户。将得到的部分用户列表作为数据源，重新绑

定，显示出来。

增加事件主要是一个跳转方法，跳转到 AddUser. aspx，如图 5-28 所示，并由 AddUser. aspx 实现增加用户功能。

图 5-28　新增用户页面（AddUser. aspx）

由系统的需求可知，管理员只可以添加新的管理员和教师，添加时必须选择两者角色之一。管理员只需要添加用户名和密码，添加完后，由管理员分配用户名和密码，其他的用户信息由使用该用户名的用户在个人信息配置模块中添加。AddUser. aspx 页面的前台 HTML 代码如程序 5-51 所示。

程序 5-51　增加用户的前台 HTML（AddUser. aspx）

```
<table>
        <tr>
          <td>选择类型：</td>
            <td><asp:RadioButtonList ID="type" runat="server" RepeatDirection=
"Horizontal">
                        <asp:ListItem Value="1">管理员</asp:ListItem>
                        <asp:ListItem Selected="True" Value="2">教师</asp:ListItem>
                  </asp:RadioButtonList></td>
          <td></td>
        </tr>
        <tr>
          <td>用户名：</td>
          <td><asp:TextBox ID="txt_UserName" runat="server"></asp:TextBox></td>
          <td><asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ErrorMessage="用户名不能为空!" ControlToValidate="txt_UserName"></asp:Required-
FieldValidator></td>
        </tr>
        <tr>
```

```
            <td>密码：</td>
            <td><asp:TextBox ID="txt_Pwd" runat="server"></asp:TextBox></td>
            <td><asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server"
 ErrorMessage="密码不能为空!" ControlToValidate="txt_Pwd"></asp:RequiredFieldValida-
tor></td>
        </tr>
        <tr>
            <td>密码确认：</td>
            <td><asp:TextBox ID="txt_Pwd1" runat="server"></asp:TextBox></td>
            <td><asp:CompareValidator ID="CompareValidator2" runat="server" ErrorMessage
="密码必须相同!" ControlToCompare="txt_Pwd" ControlToValidate="txt_Pwd1"></asp:Com-
pareValidator></td>
        </tr>
        <tr>
             <td></td>
             <td><asp:Button  ID="bt_Submit" Text="提交" runat="server" OnClick="bt_
Submit_Click"/>
                   <asp:Button ID="bt_Cancel" runat="server" Text="返回" OnClick=
"bt_Cancel_Click" /></td>
             <td></td>
        </tr>
</table>
```

其中，用到了验证控件 RequiredFieldValidator 和 CompareValidator，用于防止用户名为空和两次输入的密码不一样。当添加完毕，单击“提交”按钮，触发提交事件 bt_Submit_Click。也可以单击“返回”按钮，触发 bt_Cancel_Click 返回到用户管理首页(Default.aspx)，后台代码如程序 5-52 所示。

程序 5-52　提交事件和返回事件的后台代码(AddUser.aspx.cs)

```
protected void bt_Submit_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        int RoleID=int.Parse(this.type.SelectedValue);
        string UserName=this.txt_UserName.Text;
        string Pwd=this.txt_Pwd.Text;
        object[] Parmas=new object[]{UserName,Pwd,RoleID};
        int mark= UserManageObject.AddUser(Parmas);
        if (mark==1)
        {
            ShowAlert("添加用户成功!");
        }
        else
```

```
                {
                    ShowAlert("添加失败!");
                }
            }
        }
        protected void bt_Cancel_Click(object sender, EventArgs e)
        {
            Response.Redirect("Default.aspx");
        }
```

在提交事件时间中，首先由 Page. IsValid 判断页面是否通过验证，只有通过验证，即 Page. IsValid 返回 True，才能继续执行提交事件。UserManageObject 的 AddUser 方法为增加用户，参数为用户名、密码和角色编号。涉及到的 SQL 语句如下。

```
INSERT INTO [User](
            UserName,
            PassWord,
            RoleID
        )
        VALUES(
            @UserName,
            @PassWord,
            @RoleID
        )
```

注：由于 User 是 SQL Server 系统中的默认关键字，所以在该 SQL 语句加上中括号"[]"，以示区别。后面讲到的 SQL 语句中，加上"[]"，也是由于这个原因。

返回事件比较简单，就是重定向到用户管理首页。

用户管理模块的所有页面都在系统根目录下的 UserManage 文件夹中。

5.4.8 课程管理模块

课程管理模块是管理员才具有权限操作的模块。具有的操作权限为查询、删除、修改和增加。该模块主要由 CourseManage 文件夹下的课程管理首页（Default. aspx）、增加课程页面（AddCourse. aspx）和修改课程页面（UpdateCourse. aspx）这三个页面实现。Default. aspx如图 5-29 右边所示。

其中课程列表由 GridView 控件实现，其前台的 HTML 页面的代码如程序 5-53 所示。

程序 5-53　GridView 控件的前台 HTML 代码（Default. aspx）

```
<asp:GridView ID="GV_CourseList" runat="server" AllowPaging="True" AutoGenerateCol-
umns="False"
```



图 5-29　课程管理首页(Default.aspx)

```
        BorderWidth="1px" CellPadding="4" Width="100%" CssClass="dataTable" PageSize="
 20"
        OnRowCommand="GV_CourseList_RowCommand">
               …
        <HeaderStyle CssClass="dataHeader" Height=30px Font-Names="Arial" Horizonta-
lAlign="Center"
           BackColor="#DEF2FE"></HeaderStyle>
   <Columns>
        <asp:BoundField DataField="CourseName" HeaderText="课程名"  HeaderStyle-Font-
Bold=false/>
        <asp:BoundField DataField="UserName" HeaderText="任课教师"  HeaderStyle-Font-
Bold=false/>
        <asp:TemplateField  HeaderStyle-Font-Bold=false>
          <ItemTemplate>
          <asp:ImageButton ID="ibt_UpdateCourse"  runat="server" class="link02"
          CommandName="UpdateCourse"  CommandArgument='<%#Eval("ID") %>'
          ImageUrl="~/Image/BtnEdit.gif" />
          <asp:ImageButton ID="ibt_DeleteCourse"  runat="server" class="link02"
          CommandName="DeleteCourse" OnClientClick="return confirm('确定删除？')"
          CommandArgument='<%#Eval("ID") %>'
          ImageUrl="~/Image/BtnDelete.gif" />
        </ItemTemplate>
        </asp:TemplateField>
      </Columns>
 </asp:GridView>
```

由上面程序可知，GridView 控件是后台动态绑定的，实现代码程序 5-54 所示。

程序 5-54　课程列表绑定的后台代码(Default.aspx.cs)

```
protected void Page_Load(object sender, EventArgs e)
  {
      if (!IsPostBack)
      {
          GV_DataBind(GetCourseList(""));
      }
  }
  private DataTable GetCourseList(string CourseName)
  {
      object[] Params=new object[] { CourseName };
      DataTable dt=CourseManageObject.GetCourses(Params);
      return dt;
  }
  private void GV_DataBind(DataTable dt)
  {
      this.GV_CourseList.DataSource=dt.DefaultView;
      this.GV_CourseList.DataBind();
  }
```

注：到这里会发现，不论是页面样式、前台 HTML 代码、后台实现代码，课程管理和用户管理，还有后面将要讲到的链接管理都很类似，用的是一个 CSS 样式，统一的编程思路，程序代码和用户管理非常类似。所以在讲到课程管理和链接管理时，不再一一详述，将只是简单介绍。

CourseManageObject 是逻辑层的课程管理类，包装了与课程管理相关的方法。GetCourses 方法是获取课程列表，参数为课程名。当课程名为空时，获取全部课程；当课程名不为空，模糊查询，获取满足条件的课程。

当页面加载时，获取全部课程，所以参数为空。GetCourses 方法设计的 SQL 语句如下所示。

当参数为空时，SQL 语句如下。

```
SELECT  C.[ID] , C.CourseName , U.UserName
                  FROM Course C INNER JOIN [User]  U ON
                  C.TeacherID=U.ID
```

当参数不为空时，SQL 语句如下。

```
SELECT  C.[ID] , C.CourseName , U.UserName
                  FROM Course C INNER JOIN [User]  U ON
                  C.TeacherID=U.ID
WHERE C.CourseName LIKE  '%'+@CourseName+'%'
```

课程的删除和修改是由 GridView 控件的 RowCommand 事件完成，CommandName ="UpdateCourse"和 CommandName="DeleteCourse"指定了命令名称，分别表示修改

课程和删除课程。CommandArgument='<%#Eval("ID")%>'表明命令的参数是后台绑定的课程编号。其具体实现如程序 5-55 所示。

程序 5-55　删除和修改事件的后台代码(Default.aspx.cs)

```
protected void GV_CourseList_RowCommand(object sender, GridViewCommandEventArgs e)
        {
            string CommandName=e.CommandName;
            string CourseID=e.CommandArgument.ToString();
            switch (CommandName)
            {
                case "DeleteCourse":
                    {
                        Delete (CourseID);
                        string CourseName=this.txt_CourseName.Text.Trim();
                        GV_DataBind(GetCourseList(CourseName));
                        break;
                    }
                case "UpdateCourse":
                    {
                        Update (CourseID);
                        break;
                    }
                default:
                    {
                        break;
                    }
             }
         }
        protected void Delete (string CourseID)
        {
            object[] Parmas=new object[] { CourseID };
            CourseManageObject.DeleteCourse(Parmas);
        }
        protected void Update (string CourseID)
        {
            string url="UpdateCourse.aspx?ID="+CourseID;
            Response.Redirect(url);
        }
```

当单击删除或修改按钮时,触发 GridView 控件的 RowCommand 事件 GV_CourseList_RowCommand。根据 CommandName 命令,执行 Delete 或 Update 方法。

Delete 方法主要由 CourseManageObject 类的 DeleteCourse 方法完成。DeleteCourse 方法为删除指定课程,参数为课程编号。Delete 方法执行完后,会重新绑定,显示

删除后的课程列表。DeleteCourse 方法涉及的 SQL 语句如下。

```
DELETE FROM Course WHERE [id]=@CourseID
```

Update 方法比较简单，将当前页面跳转到修改课程页面 UpdateCourse.aspx，如图 5-30 所示，并传递一个参数 ID，值为当前修改的课程编号。

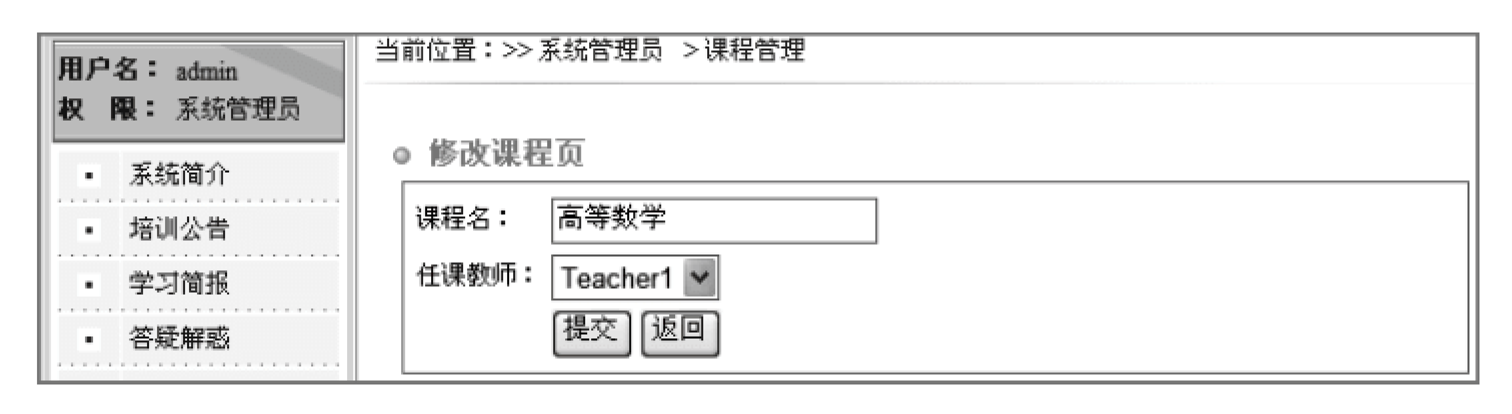


图 5-30 修改课程页面(UpdateCourse.aspx)

修改课程页面的前台 HTML 代码如程序 5-56 所示。

程序 5-56 修改课程页面的前台 HTML 代码(UpdateCourse.aspx)

```
    <table>
      <tr>
        <td>课程名：</td>
        <td><asp:TextBox ID="txt_CourseName" runat="server"></asp:TextBox></td>
        <td><asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ErrorMessage ="课程名不能为空!" ControlToValidate =" txt _ CourseName " >
</asp:RequiredFieldValidator></td>
      </tr>
      <tr>
        <td>任课教师：</td>
        <td><asp:DropDownList ID="ddl_Teacher" runat="server"></asp:DropDownList>
</td>
        <td></td>
      </tr>
      <tr>
         <td></td>
          <td><asp:Button  ID="bt_Submit" Text="提交" runat="server" OnClick=
"bt_Submit_Click"/>
             <asp:Button ID="bt_Cancel" runat="server" Text="返回" OnClick="bt_Can-
cel_Click"/></td>
         <td></td>
      </tr>
</table>
```

如程序 5-56 所示，有一个验证控件 RequiredFieldValidator，用于限制用户名不能为空。修改课程页面的后台代码实现如程序 5-57 所示。

程序 5-57 修改课程页面的后台代码(UpdateCourse.aspx.cs)

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        ddlDatabind();
        int CourseID=int.Parse(Request.QueryString["ID"].ToString());
        object[] Parmas=new object[] { CourseID };
        DataTable dt=CourseManageObject.GetCoursesbyID(Parmas);
        this.txt_CourseName.Text=dt.Rows[0]["CourseName"].ToString();
        this.ddl_Teacher.SelectedValue=dt.Rows[0]["TeacherID"].ToString();
    }
}
protected void ddlDatabind()
{
    DataTable dt=UserManageObject.GetTeachers();
    this.ddl_Teacher.DataSource=dt.DefaultView;
    this.ddl_Teacher.DataTextField="UserName";
    this.ddl_Teacher.DataValueField="ID";
    this.ddl_Teacher.DataBind();
}
protected void bt_Submit_Click(object sender, EventArgs e)
{
    string CourseName=this.txt_CourseName.Text.Trim();
    int TeacherID=int.Parse(this.ddl_Teacher.SelectedValue);
    int CourseID=int.Parse(Request.QueryString["ID"].ToString());
    object[] Parmas=new object[] { CourseID, CourseName, TeacherID };
    int mark=CourseManageObject.UpdateCourse(Parmas);
    if (mark<0){
                ShowAlert("修改失败!"); }
    else {
        ShowAlert("修改成功!");           }
}
protected void bt_Cancel_Click(object sender, EventArgs e)
{
    Response.Redirect("Default.aspx");
}
```

如程序 5-57 所示，页面第一次加载时，首先执行 ddlDatabind 方法，该方法通过 UserManageObject 类的 GetTeachers 从数据库中获取教师列表，并以获取的列表作为下拉列表 ddl_Teacher 的数据源，进行绑定。GetTeachers 方法涉及的 SQL 语句如下所示。

```
SELECT U.[ID],U.UserName FROM [User] U
                    INNER JOIN Role R  ON  U.RoleID=R.RoleID
                    WHERE RoleName like '%教师%'
```

然后获取查询字符串课程编号(ID)的值,并以该课程编号为参数,执行 CourseManageObject 类的 GetCoursesbyID 方法,获取该指定的课程信息,初始化课程名称和任课教师。GetCoursesbyID 方法设计的 SQL 语句如下所示。

```
SELECT C.CourseName,C.TeacherID,U.UserName FROM Course C
                    INNER JOIN [User] U ON C.TeacherID=U.[ID]
                    WHERE C.[ID]=@CourseID
```

当单击“提交”按钮,触发 bt_Submit_Click 事件,该事件主要是完成修改功能,由 CourseManageObject 的 UpdateCourse 方法实现,该方法的参数为 CourseID、CourseName 和 TeacherID。设计到的 SQL 语句如下所示。

```
UPDATE Course
      SET  TeacherID=@TeacherID ,CourseName=@CourseName
  WHERE ID=@CourseID
```

当单击“返回”按钮,触发 bt_Cancel_Click 事件,返回到课程管理首页。

在图 5-29 所示的课程管理页面中,还有查询和新增课程功能,其前台 HTML 的页面如程序 5-58 所示。

程序 5-58　查询和新增课程的前台 HTML 代码(Default.aspx)

```
<table width="100%">
  <tr>
      <td align="right" width="15%">课程名:</td>
      <td align="left" width="15%">
        <asp:TextBox ID="txt_CourseName" runat="server"></asp:TextBox></td>
      <td align="left" width="15%">
        <asp:Button ID="select" runat="server" Text="查询"  OnClick="select_
Click" /></td>
      <td align="left" width="15%">
        <asp:Button ID="Add" runat="server" Text="新增课程" OnClick="Add_Click" /
></td>
     </tr>
</table>
```

单击查询和新增课程,触发 select_Click 和 Add_Click 事件,后台代码如程序 5-59 所示。

程序 5-59　查询和新增课程的后台代码(Default.aspx.cs)

```
protected void select_Click(object sender, EventArgs e)
{
    string CourseName=this.txt_CourseName.Text.Trim();
    GV_DataBind(GetCourseList(CourseName));
```

```
    }
    protected void Add_Click(object sender, EventArgs e)
    {
        string url="AddCourse.aspx";
        Response.Redirect(url);
    }
```

查询事件跟课程列表初始化类似，只不过这里课程名不为空，GetCourseList 方法调用有参数的 GetCourses 方法，得到满足条件的课程列表，再进行 GridView 的绑定。

新增课程事件很简单，将当前页面跳转到增加课程页面(AddCourse.aspx)，如图 5-31 所示，由增加课程页面完成新增课程功能。

图 5-31　增加课程页面(AddCourse.aspx)

增加课程页面的前台 HTML 代码跟修改类似，只是提交事件不同，如程序 5-60 所示。

程序 5-60　提交事件的后台代码(AddCourse.aspx.cs)

```
protected void bt_Submit_Click(object sender, EventArgs e)
{
    string CourseName=this.txt_CourseName.Text.Trim() ;
    string TeacherID=this.ddl_Teacher.SelectedValue ;
    object[] Parmas=new object[] { CourseName, TeacherID };
    int mark=CourseManageObject.AddCourse(Parmas);
    if (mark==1){
        ShowAlert("添加课程成功!");   }
    else{
        ShowAlert("添加失败!");        }
}
```

提交事件主要由 CourseManageObject 类的 AddCourse 方法完成增加课程功能，该方法的参数是课程名称(CourseName)、任课教师编号(TeacherID)。涉及的 SQL 语句如下所示。

```
INSERT INTO Course(
                    CourseName,
                    TeacherID
```

```
            )
        VALUES (
                    @CourseName,
                    @TeacherID
            )
```

返回事件比较简单,就是跳转到课程管理首页。

课程管理模块的所有页面都在系统根目录下的 CourseManage 文件夹中。

5.4.9 作业展示模块(教师)

作业展示模块是针对教师和学员而言的,当用户以教师或者学员的身份登录时就会在管理平台看见作业展示模块。教师和学员所对应的作业展示模块不同,以教师身份登录时,作业展示模块具有教师发布新的作业、修改作业、删除作业及批改学员作业的功能。

首先,看一下以教师身份登录时作业展示的页面,如图 5-32 所示。

当前位置:>> 教师 > 作业展示

请先选择课程,再点击按钮!

语文 [查询作业] [新增作业]

课程名称	教师姓名	作业标题	发布日期	修改	删除
语文	教师1	远程教育的特点	2008-1-2 16:25:10	修改	删除
语文	教师1	远程教育的概述	2008-1-3 15:23:00	修改	删除
语文	教师1	远程教育的趋势	2008-1-4 17:45:00	修改	删除

< >

图 5-32 作业展示页面(教师)

从图 5-32 可以看出,当教师选择课程后,单击“查询作业”按钮时会看到该教师自己针对该课程所发布的作业,包括课程名称、教师姓名、作业标题、发布日期、修改和删除 6 部分。要实现这样的效果,其前台代码片段如程序 5-61 所示。

程序 5-61 作业展示页面的 HTML 代码(TeacherHomework.aspx)

```
<table runat="server" style="width: 100%; height: 86px" id="table">
  <tr>
    <td colspan="2" style="height: 26px; width: 100%;">
      <asp:DropDownList ID="Course" runat="server"  AutoPostBack="true"/>
       <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text=
"查询作业" />
      <asp:Button ID="Button2" runat="server" Text="新增作业" OnClick="Button2_Click"
/></td></tr>
  <tr id="lwwai" runat="server">
    <td colspan="4">
```

```
      <asp:DataGrid ID="DataGrid1" runat="server" AllowPaging="True" AutoGenerateCol-
umns="False"
           BorderWidth="1px" CellPadding="4" Width="100%"
           CssClass="dataTable" PageSize="20" >
        <AlternatingItemStyle CssClass="DgAltItem" BackColor=#EEEEEE />
          <ItemStyle CssClass="DgItem" HorizontalAlign="Center" BackColor=White />
          <PagerStyle ForeColor="DarkGray" />
          <HeaderStyle CssClass="dataHeader" Height=30px Font-Names="Arial" Horizon-
talAlign="Center" BackColor="#DEF2FE"></HeaderStyle>
          <Columns>
              < asp: BoundColumn  HeaderText =" 课 程 名 称 "  DataField =" CourseName " >
</asp:BoundColumn>
            <asp:BoundColumn HeaderText="教师姓名" DataField="RealName"></asp:Bound-
Column>
            <asp:TemplateColumn HeaderText="作业标题">
              <ItemTemplate>
               <asp:HyperLink Text='<%#DataBinder.Eval(Container, "DataItem.Title")%
>' NavigateUrl= '<% # myfunc2 (DataBinder. Eval (Container. DataItem, "HomeWorkID"))% > '
Runat="server" Target="_blank" ID="Hyperlink1">
               </asp:HyperLink>
              </ItemTemplate>
           </asp:TemplateColumn>
           <asp: BoundColumn  HeaderText =" 发 布 日 期 "  DataField =" PublishTime " >
</asp:BoundColumn>
              <asp:TemplateColumn HeaderText="修改">
                <ItemTemplate>
                 <asp:HyperLink Text='修改' NavigateUrl='<%#myfunc3(DataBinder.Eval
(Container.DataItem,"HomeWorkID"))%>' Runat="server" Target="_blank"
           ID="Hyperlink2">
                </asp:HyperLink>
              </ItemTemplate>
             </asp:TemplateColumn>
       < asp: ButtonColumn  CommandName =" Delete "  HeaderText =" 删 除 "  Text =" 删 除 " >
</asp:ButtonColumn>
      </Columns>
     </asp:DataGrid>
…
```

程序 5-62 实现了 DropDownList 下拉列表中课程的绑定。

程序 5-62　作业展示页面中的课程绑定（TeacherHomework. aspx. cs）

```
public void coursebind()
        {
```

```
                DataTable CourseTable=new DataTable();
                 CourseTable= IndexObject.GetCoursebyTeacherID(Convert.ToInt32(Session["
                 TeacherID"]));
                Course.DataSource=CourseTable;
                Course.DataTextField="CourseName";
                Course.DataValueField="ID";
                Course.DataBind();
                Course.Items.Insert(0, new ListItem("--选择课程--", "0"));
    }
```

在程序5-62中通过调用IndexObject的方法GetCoursebyTeacherID来获得该教师所教的课程。GetCoursebyTeacherID方法所涉及的SQL语句如下所示。

```
select CourseName,ID
from Course
where TeacherID=@TeacherID
```

当教师选择课程后，单击“查询作业”按钮时，会显示该教师所发布的作业，如程序5-63所示。

程序5-63　作业展示页面作业绑定(TeacherHomework.aspx.cs)

```
public void coursebindbyTeacherIDandcourseID()
  {
      int CourseID=Convert.ToInt32(Course.SelectedValue.ToString());
      DataTable CourseTable1=new DataTable();
      CourseTable1=
      IndexObject.GetcoursebindbyTeacherIDandcourseID(Convert.ToInt32(Session["Teach-
erID"]), CourseID);
      DataGrid1.DataSource=CourseTable1;
      DataGrid1.DataBind();
  }
```

在程序5-63中通过调用IndexObject的方法GetcoursebindbyTeacherIDandcourseID来获得该教师对所选课程发布的作业。GetcoursebindbyTeacherIDandcourseID方法所涉及的SQL语句如下所示。

```
select CourseName,RealName,Title,PublishTime,HomeWork.ID as HomeWorkID
from HomeWork inner join [User] on HomeWork.TeacherID= [User].ID
              inner join Course on Course.ID= HomeWork.CourseID
where  HomeWork.TeacherID=@TeacherID and HomeWork.CourseID=@CourseID
```

单击每个作业后面的“修改”按钮，会跳转到作业修改页，如图5-33所示。

从图5-33中，可以很容易地看出，它是由作业标题、作业要求、提交和取消4个部分组成的。要实现这样的效果，其前台代码片段如程序5-64所示。

作业标题：

远程教育的特点

作业要求：

概述远程教育的特点

提交　取消

图 5-33　作业修改页面

程序 5-64　作业修改页面的 HTML 代码(EditTeacherHomework.aspx)

```
<table border="1" style="border-color:Blue; width:600px;" cellpadding="0" cellspacing="0">
     <tr>
        <td  align="left" style="height: 40px">
          <asp:Label ID="title" runat="server" Font-Bold="True" Text="作业标题："></asp:Label>
        </td>
     </tr>
     <tr>
       <td align="left" style="height: 60px">
         <asp:TextBox ID="TitleContent" runat="server" Height="60px" TextMode="MultiLine" Width="600px"></asp:TextBox>
       </td>
     </tr>
     <tr>
       <td align="left" style="height: 40px">
         <asp:Label ID="Label1" runat="server" Text="作业要求：" Font-Bold="True"></asp:Label>
       </td>
     </tr>
     <tr>
       <td align="left" style="height: 120px">
         <asp:TextBox ID="HomeWorkContent" runat="server" Height="120px" TextMode="MultiLine" Width="600px"></asp:TextBox>
       </td>
     </tr>
     <tr>
       <td>
```

```
            <asp:Button ID="Button1" runat="server" Text="提交" OnClick="Button1_Click"
    />
           <asp:Button ID="Button2" runat="server" Text="取消" />
         </td>
       </tr>
     </table>
```

页面初始化程序如程序 5-65 所示。

程序 5-65　作业修改页面初始化代码（EditTeacherHomework. aspx. cs）

```
protected void Page_Load(object sender, EventArgs e)
    {
            if (! IsPostBack)
            {
                     Session [ " HomeWorkID "] = Convert. ToInt32 (Request. QueryString
["HomeWorkID"]);
                 DataTable ContentTable=new DataTable();
                 ContentTable= IndexObject.GetTeacherHomeworkContentbyHomeWorkID (Convert.
ToInt32(Session["HomeWorkID"]));
                 string homeworktitle= ContentTable.Rows[0]["HomeWorkTitle"].ToString
();
                 TitleContent.Text=homeworktitle;//作业的标题
                 stringteachercontent= ContentTable.Rows [0] ["HomeWorkContent"].ToS-
tring();
                HomeWorkContent.Text=teachercontent;//老师布置作业的要求
            }
   }
```

在程序 5-65 中通过调用 IndexObject 的方法 GetTeacherHomeworkContentby-HomeWorkID 来获得该教师对所选课程发布的作业标题及内容要求。

GetTeacherHomeworkContentbyHomeWorkID 方法所涉及的 SQL 语句如下所示。

```
select HomeWork.Title as HomeWorkTitle,HomeWork.Content as HomeWorkContent
from HomeWork
where HomeWork.ID=@HomeWorkID
```

修改完毕，当单击“提交”按钮时会触发 Click 事件，该事件处理程序如程序 5-66 所示。

程序 5-66　作业修改页面保存修改信息代码(EditTeacherHomework. aspx. cs)

```
      protected void Button1_Click(object sender, EventArgs e)
        {
            try
             {
```

```
                    string titlecontent=TitleContent.Text;
                    string homeworkcontent=HomeWorkContent.Text;
                    IndexObject.UpdateTeacherHomeworkTitleAndContent(Convert.ToInt32(Session
["HomeWorkID"]), titlecontent, homeworkcontent);
                    Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "asdf", "
alert('提交成功!');window.close();", true);
                    Response.Write("<script>parent.opener=null;parent.close()
</script> ");
                }
                catch
                {
                    Page.ClientScript.RegisterClientScriptBlock(this.GetType(),
"asdf", "alert('提交未成功,请重新修改!');", true);
                    Response.Write("<script>parent.opener=null;parent.close()
</script>");
                }
            }
```

在程序 5-66 中通过调用 IndexObject 的方法 UpdateTeacherHomeworkTitleAndContent 来更新该教师对所选课程发布的作业标题及内容要求。

UpdateTeacherHomeworkTitleAndContent 方法所涉及的 SQL 语句如下所示。

```
update HomeWork
set HomeWork.Title=@titlecontent ,
    HomeWork.Content=@homeworkcontent
where HomeWork.ID=@HomeWorkID
```

作业修改页面的完整源程序在\WEB\Teacher\ EditTeacherHomework.aspx 和\WEB\Teacher\ EditTeacherHomework.aspx.cs 文件中。

单击每个作业后面的“删除”按钮,会删除该条作业,如程序 5-67 所示。

程序 5-67　删除作业代码(TeacherHomework.aspx.cs)

```
protected void DataGrid1_DeleteCommand(object source, DataGridCommandEventArgs e)
        {
            try
            {
                int HomeWorkID=Convert.ToInt32(DataGrid1.DataKeys[e.Item.ItemIndex]);
                IndexObject.DeleteHomeWork(HomeWorkID);
                Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "asdf","
alert('删除成功!');window.close();", true);
                Response.Write("<script>parent.opener=null;parent.close()
</script>");
```

```
            }
            catch
            {
                Page. ClientScript. RegisterClientScriptBlock (this. GetType ( ),
"asdf", "alert('删除不成功!');", true);
                Response. Write ("< script > parent. opener = null; parent. close ()
</script>");
            }
    }
```

在程序 5-67 中通过调用 IndexObject 的方法 DeleteHomeWork 来删除该条作业。DeleteHomeWork 方法所涉及的 SQL 语句如下所示。

```
delete from HomeWork where ID=@HomeWorkID
delete from Answer where HomeWorkID=@HomeWorkID
```

作业展示页面的完整源程序在\WEB\Teacher\ TeacherHomework. aspx 和\WEB\Teacher\ TeacherHomework. aspx. cs 文件中。

单击作业的标题会跳转到学生回答作业列表的页面，如图 5-34 所示。

做作业的情况如下：

学号	学生姓名	查看作业
13	张三	远程教育的特点
14	李四	远程教育的特点

图 5-34　学生回答作业列表

从图 5-34 可以看出，在该页面上列出了学生回答该条作业的情况，包括学生的学号、学生的姓名及查看作业三个部分。要实现这样的效果，其代码片段如程序 5-68 和程序 5-69 所示。

程序 5-68　学生回答作业列表的 HTML 代码(HomeWorkReply. aspx)

```
<asp:Label ID="Label1" runat="server" Text="做作业的情况如下：></asp:Label> <
/div>
<asp:DataGrid ID="DataGrid1" runat="server" AutoGenerateColumns="False" CssClass="
dataTable">
    <AlternatingItemStyle CssClass="DgAltItem" BackColor=#EEEEEE />
    <ItemStyle CssClass="DgItem" HorizontalAlign="Center" BackColor=White />
    <PagerStyle ForeColor="DarkGray" />
    <HeaderStyle CssClass="dataHeader" Height=30px Font-Names="Arial" Horizonta-
lAlign="Center" BackColor="#DEF2FE"></HeaderStyle>
    <Columns>
      <asp:BoundColumn HeaderText="学号" DataField="StudentID"></asp:BoundColumn>
      <asp:BoundColumn HeaderText="学生姓名" DataField="StudentName"></asp:Bound-
Column>
```

```
            <asp:TemplateColumn HeaderText="查看作业">
              <ItemTemplate>
                <asp:HyperLink Text='<%#DataBinder.Eval(Container, "DataItem.HomeWork-
Title")%>' NavigateUrl='<%#myfunc2(DataBinder.Eval(Container.DataItem,"AnswerID"))%
>' Runat="server" Target="_blank" ID="Hyperlink1"   >
                  </asp:HyperLink>
                </ItemTemplate>
              </asp:TemplateColumn>
          </Columns>
</asp:DataGrid>
```

程序 5-69　学生回答作业列表的后台代码(HomeWorkReply.aspx.cs)

```
public void studentbind()
  {
              DataTable CourseTable1=new DataTable();
              CourseTable1=IndexObject.GetAnswerListbyHomeWorkID(Convert.ToInt32(Session["
HomeworkID"]));
              DataGrid1.DataSource=CourseTable1;
              DataGrid1.DataBind();
    }
```

在程序 5-69 中通过调用 IndexObject 的方法 GetAnswerListbyHomeWorkID 来获得学生回答该条作业的列表。GetAnswerListbyHomeWorkID 方法所涉及的 SQL 语句如下所示。

```
select HomeWork.Title as HomeWorkTitle, [User].ID as StudentID, [User].RealName as Stu-
dentName,Answer.ID as AnswerID
from [User] inner join Answer on [User].ID=Answer.StudentID
      inner join HomeWork on Answer.HomeworkID=HomeWork.ID
where Answer.HomeworkID=@HomeworkID
```

学生回答作业列表的完整源程序在\WEB\Teacher\ HomeWorkReply.aspx 和\WEB\Teacher\ HomeWorkReply.aspx.cs 文件中。

单击图 5-34 中作业标题时会跳转到教师批改作业的页面，如图 5-35 所示。

从图 5-35 可以看出，此页面主要由作业标题、学号、姓名、老师布置作业的要求、学生回答作业的内容、教师的评语、得分、提交和取消 9 个部分组成。要实现这样的效果，其前台代码片断如程序 5-70 所示。

程序 5-70　教师批改作业页面的 HTML 代码(StudentAnswer.aspx)

```
<table border="1" style="border-color:Blue; width:600px;" cellpadding="0" cellspacing
="0">
   <tr>
    <td  align="left" style="height: 40px">
```

远程教育

学号： 13　　姓名： 张三

作业要求：

远程教育概述

作业内容：

远程教育是指通过音频、视频（直播或录像）及包括实时和非实时在内的计算机技术把培训课程传送到远处（校园外）的教育。这一远程教育的定义不包括专门在校园内及只通过书面形式进行的课程（尽管远程教育中也包括一些在校的指导或考试，其中一些指导也有可能是通过书面来进行的）。

教师评语：

得分：

提交　取消

图 5-35　教师批改作业的页面

```
<asp:Label ID="lable1" Text="学号：" runat="server" Font-Bold="True"></asp:Label>
      <asp:Label ID="id" runat="server" Font-Underline="False"></asp:Label>
                        
      <asp:Label ID="label2" Text="姓名：" runat="server" Font-Bold="True"></asp:Label>
      <asp:Label ID="name" runat="server" Font-Underline="False"></asp:Label></td>
  </tr>
  <tr>
   <td align="left" style="height: 40px">
      <asp:Label ID="Label1" runat="server" Text="作业要求：" Font-Bold="True"></asp:Label>
   </td>
  </tr>
  <tr>
   <td align="left" style="height: 120px">
          <asp:TextBox ID="homeworkcontent" runat="server" Height="120px" TextMode="MultiLine" Width="600px" ReadOnly="True"></asp:TextBox></td>
  </tr>
```

```
    <tr>
     <td align="left" style="height: 40px">
       <asp:Label ID="Label2" runat="server" Text="作业内容: " Font-Bold="True"></
asp:Label>
     </td>
    </tr>
    <tr>
     <td align="left" style="height: 300px">
       <asp:TextBox ID="content" runat="server" TextMode="MultiLine" Width="600px"
Height="300px" ReadOnly="True"></asp:TextBox></td>
    </tr>
    <tr>
     <td align="left" style="height: 40px">
       <asp:Label ID="Label3" runat="server" Text="教师评语: " Font-Bold="True"></
asp:Label>
     </td>
    </tr>
    <tr>
     <td align="left" style="height: 100px">
       <asp:TextBox ID="pingyu" runat="server" Height="100px" TextMode="MultiLine"
Width="600px"></asp:TextBox></td>
    </tr>
    <tr>
     <td align="left" style="height: 40px">
      <asp:Label ID="Label4" runat="server" Text="得 分: " Font-Bold="True">
</asp:Label></td>
    </tr>
    <tr>
     <td align="left" style="height: 50px">
          <asp:TextBox ID="grade" runat="server" Height="50px" TextMode="MultiLine"
Width="600px"></asp:TextBox></td>
    </tr>
    <tr>
     <td>
       <asp:Button ID="Button1" runat="server" Text="提交" OnClick="Button1_Click" />
       <asp:Button ID="Button2" runat="server" Text="取消" OnClick="Button2_Click" />
</td>
    </tr>
</table>
```

通过接收学生回答作业的参数 AnswerID 来初始化页面信息，包括作业标题、学号、姓名、作业要求及学生回答作业的内容等几个部分，如程序 5-71 所示。

程序 5-71　教师批改作业页面的初始化代码(StudentAnswer.aspx.cs)

```
protected void Page_Load(object sender, EventArgs e)
  {
    Session["AnswerID"]=Convert.ToInt32(Request.QueryString["AnswerID"]);
    DataTable ContentTable=new DataTable();
    ContentTable=IndexObject.GetAnswerContentbyAnswerID(Convert.ToInt32(Session["An-
swerID"]));
    string homeworktitle=ContentTable.Rows[0]["HomeWorkTitle"].ToString();
    title.Text=homeworktitle;                          //作业的标题
    string studentid=ContentTable.Rows[0]["StudentID"].ToString();
    id.Text=studentid;                                 //学生的学号
    string studentname=ContentTable.Rows[0]["StudentName"].ToString();
    name.Text=studentname;                             //学生的姓名
    string teachercontent=ContentTable.Rows[0]["HomeWorkContent"].ToString();
    homeworkcontent.Text=teachercontent;               //老师布置作业的要求
    string answercontent=ContentTable.Rows[0]["AnswerContent"].ToString();
    content.Text=answercontent;                        //学生回答作业的内容
 }
```

在程序 5-71 中通过调用 IndexObject 的方法 GetAnswerContentbyAnswerID 来获得学生回答该条作业的相关内容。GetAnswerContentbyAnswerID 方法所涉及的 SQL 语句如下所示。

```
select HomeWork.Title as HomeWorkTitle,[User].ID as StudentID,[User].RealName as
      StudentName,HomeWork.Content as HomeWorkContent,Answer.Content as
      AnswerContent,Answer.TeacherComment as TeacherComment,Answer.HomeworkS as Home-
workS
from [User] inner join Answer on [User].ID=Answer.StudentID
          inner join HomeWork on HomeWork.ID=Answer.HomeworkID
where Answer.ID=@AnswerID
```

批改作业的提交操作实际上是一个典型的数据库 Update 操作，如程序 5-72 所示。

程序 5-72　教师批改作业页面的提交代码(StudentAnswer.aspx.cs)

```
protected void Button1_Click(object sender, EventArgs e)
  {
    try
      {
         string teacherpingyu=pingyu.Text;
         float studentgrade=Convert.ToInt32(grade.Text.Trim());
         IndexObject.UpdateTeacherCommentAndHomeworkS (teacherpingyu, studentgrade,
Convert.ToInt32(Session["AnswerID"]));
```

```
                Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "asdf", "alert('
提交成功!');window.close();", true);
                Response.Write("<script>parent.opener=null;parent.close()</script>");
            }
        catch
            {
                Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "asdf", "alert('
提交未成功,请重新修改!');", true);
                Response.Write("<script>parent.opener=null;parent.close()</script>");
                }
    }
```

在程序 5-72 中通过调用 IndexObject 的方法 UpdateTeacherCommentAndHomeworkS 把教师对学生作业的评语及作业的分数写入数据库中。UpdateTeacherCommentAndHomeworkS 方法所涉及的 SQL 语句如下所示。

```
update Answer
set TeacherComment=@TeacherComment,
     HomeworkS=@HomeworkS
where Answer.ID=@AnswerID
```

教师批改作业页面的完整源程序在\WEB\Teacher\ StudentAnswer.aspx 和\WEB\Teacher\ StudentAnswer.aspx.cs 文件中。

到此为止,单击"查询作业"按钮后相关的操作已经介绍完毕。从图 5-32 可以看出,当教师选择课程后,单击"新增作业"按钮时会跳转到新增作业页面,如图 5-36 所示。

图 5-36　教师新增作业页面

该教师可以对该课程发布新的作业,包括所属课程、教师姓名、作业标题和发布作业的内容 4 个部分。要实现这样的效果,其前台代码片断如程序 5-73 所示。

程序 5-73　教师新增作业页面的 HTML 代码(AddTeacherHomeWork.aspx)

```
<table border="1" style="border-color:Blue; width:600px;" cellpadding="0" cellspacing
="0">
    <tr valign="middle">
     <td align="center" style="height: 50px" colspan="4">
           <asp:Label ID="title" runat="server" Font-Bold="True"  ForeColor="Black"
Text="新增作业"></asp:Label></td>
    </tr>
    <tr>
     <td  style="height: 40px">
          <asp:Label ID="label1" Text="所属课程：" runat="server" Font-Bold="True">
</asp:Label>
     </td>
     <td  style="height: 40px; width: 100px;">
          < asp:TextBox ID="coursename" runat="server" Height="40px" Width="120px"
ReadOnly="True"></asp:TextBox></td>
     <td  style="height: 40px">
       <asp:Label ID="label2" Text="教师姓名：" runat="server" Font-Bold="True"></
asp:Label>
    </td>
    <td  style="height: 40px; width: 100px;">
         < asp:TextBox ID="teachername" runat="server" Width="120px" Height="40px"
ReadOnly="True"></asp:TextBox></td>
   </tr>
   <tr>
    <td align="left" style="height: 40px" colspan="4">
      <asp:Label ID="Label1" runat="server" Text="作业标题：" Font-Bold="True"></
asp:Label>
    </td>
   </tr>
   <tr>
    <td align="left" style="height: 120px" colspan="4">
          < asp:TextBox ID="homeworktitle" runat="server" Height="120px" TextMode="
MultiLine" Width="600px" ></asp:TextBox></td>
   </tr>
   <tr>
    <td align="left" style="height: 40px" colspan="4">
      <asp:Label ID="Label2" runat="server" Text="作业内容：" Font-Bold="True"></
asp:Label>
    </td>
   </tr>
```

```
    <tr>
     <td align="left" style="height: 300px" colspan="4">
           <asp:TextBox ID="homeworkcontent" runat="server" TextMode="MultiLine" Width
="600px" Height="300px" ></asp:TextBox></td>
    </tr>
    <tr>
     <td colspan="4">
           <asp:Button ID="Button1" runat="server" Text="提交" OnClick="Button1_Click"
  />
           <asp:Button ID="Button2" runat="server" Text="取消" OnClick="Button2_Click"
  /></td>
    </tr>
</table>
```

后台代码类似于教师批改作业的页面,先初始化显示教师所选的课程名称及教师的姓名,这两项是不能修改的。通过调用 IndexObject 的方法 GetTeacherNamebyTeacherID 获得教师的姓名,通过调用 IndexObject 的方法 GetCourseNameByCourseID 获得课程的名称。对于单击“提交”按钮,实际上就是一个典型的数据库 Insert 操作,通过调用 IndexObject 方法的 InsertHomeWork 完成新增作业的功能。这里具体的代码就不详细说明了。GetTeacherNamebyTeacherID、GetCourseNameByCourseID 和 InsertHomeWork 所涉及的 SQL 语句如下所示。

```
select RealName as TeacherName    //获得教师姓名
from [User]
where [User].ID=@TeacherID
```

```
select CourseName    //获得课程名称
from Course
where ID=@CourseID
```

```
declare @CourseID int
set @CourseID=(select ID from Course where CourseName=@CourseName)
declare @ColumnID int
set @ColumnID=(select ID from [Column] where ColumnName='作业展示')
insert into HomeWork(ColumnID,CourseID,TeacherID,Title,Content,PublishTime)    //插入
操作
values(@ColumnID,@CourseID,@TeacherID,@HomeWorkTitle,@HomeWorkContent,getdate())
```

教师新增作业页面的完整源程序在\WEB\Teacher\ AddTeacherHomeWork.aspx 和\WEB\Teacher\ AddTeacherHomeWork.aspx.cs 文件中。

作业展示模块的完整源程序在\WEB\Teacher 文件夹中。

5.4.10 作业展示模块(学员)

如前所述的作业展示模块(教师),教师布置完作业后,学员就需要对作业进行回答,这就是学员作业展示模块的功能。

学员作业展示模块的主页面如图 5-37 所示。

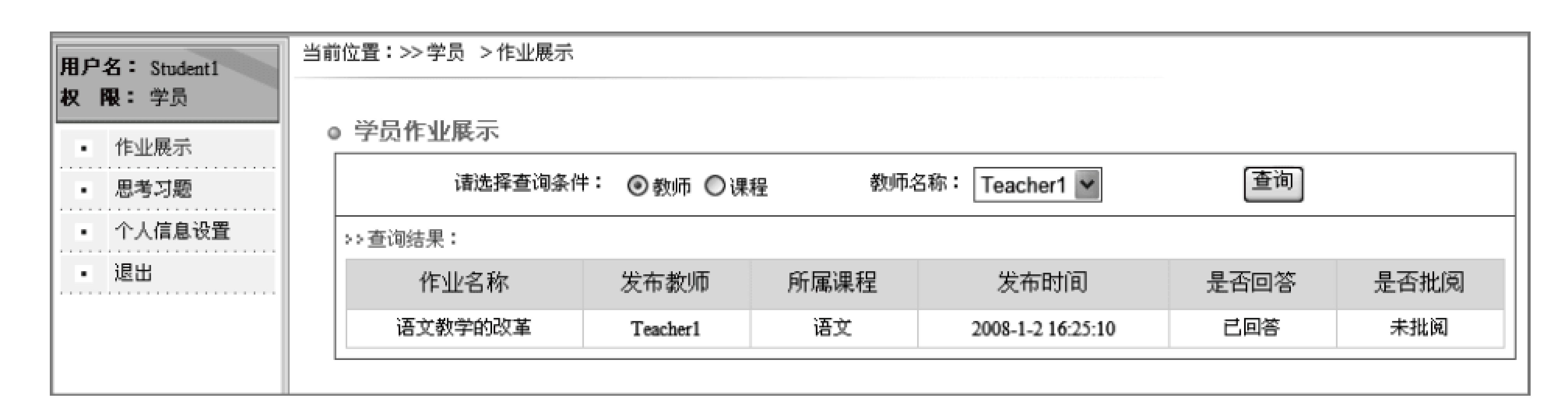


图 5-37 学员作业展示模块的主页面(Default.aspx)

在图 5-37 中,学员登录后,可以看到教师布置的作业及学员回答的情况。当教师布置的作业,学员没有回答,在“是否回答”列上显示“未回答”,反之则显示“已回答”;当学员回答完作业后,等待教师批阅,若没有批阅,则“是否批阅”列显示“未评阅”,反之则显示“已批阅”。学员做作业或者查看教师批阅情况,直接单击“作业名称”列下的标题,即可进行答题或者查看批阅情况。作业列表的前台 HTML 代码如程序 5-74 所示。

程序 5-74 作业列表的前台 HTML 代码(Default.aspx)

```
<asp:GridView ID="GV_HomeWorkList" runat="server" AllowPaging="True" AutoGenerateColumns="False" CssClass="dataTable" PageSize="20" >
        <AlternatingRowStyle CssClass="DgAltItem" BackColor=#EEEEEE></AlternatingRowStyle>
                    …     (与前面的 GridView 控件的样式类似)
        <HeaderStyle CssClass="dataHeader" Height=30px Font-Names="Arial" HorizontalAlign="Center" BackColor="#DEF2FE"></HeaderStyle>
        <Columns>
          <asp:TemplateField HeaderText="作业名称">
            <ItemTemplate>
            <a id="A1" href='<%#"Transfer.aspx?ColumnId="+DataBinder.Eval(Container.DataItem,
                "ColumnID")+"&HomeworkID="+DataBinder.Eval(Container.DataItem,
 "HomeworkID")%>' class="navigator"  runat="server"><%#DataBinder.Eval(Container.
 DataItem, "Title")%>
            </a>
            </ItemTemplate>
          </asp:TemplateField>
          <asp:BoundField DataField="UserName" HeaderText="发布教师" />
          <asp:BoundField DataField="CourseName" HeaderText="所属课程"  />
```

```
            <asp:BoundField DataField="PublishTime" HeaderText="发布时间"  />
            <asp:BoundField DataField="IsAnswer" HeaderText="是否回答"  />
            <asp:BoundField DataField="IsReply" HeaderText="是否批阅"  />
        </Columns>
    </asp:GridView>
```

如程序所示,"作业名称"列是一个模板列,包含一个超级链接,链接的地址为 Transfer.aspx 页面,在稍后会讲到,传递的参数是后台绑定的。其他的列都是绑定列。作业列表后台绑定代码如程序 5-75 所示。

程序 5-75　作业列表(GridView)的后台绑定(Default.aspx.cs)

```
string StudentID=((UserData)Session["Current_User"]).UserID;
string Parmas=new object[] { "", "", StudentID };
dt=IndexObject.GetHomeWorks(Parmas);
GV_HomeWorkList.DataSource=dt.DefaultView;
GV_HomeWorkList.DataBind();
```

得到作业列表是由 IndexObject 类的 GetHomeWorks 方法,该方法一共有三个参数,第一个参数是课程编号,第二个参数是教师编号,第三个参数是学员编号。该处的程序思想跟前面讲到的用户管理和课程管理类似,初始化列表和查询共用一个方法,这里 IndexObject 类的 GetHomeWorks 方法,前面两个参数是给查询用。当前两个参数都为空字符串时,表明是初始化作业列表;当不为空时,表明是根据某个条件查询,得到满足条件的作业列表,这个在稍后会讲到。所以在这里,前两个参数都为空字符串,第三个参数是学员编号,可以从 Session["Current_User"]获得。得到作业列表后,对 ID 为 GV_HomeWorkList 的 Gridview 进行绑定(注:学员登录时,学员的基本信息会存放在 Session["Current_User"]中)。

GetHomeWorks 涉及的 SQL 语句如下所示。

```
Select H.[ID] AS HomeworkID, H.ColumnID, H.CourseID, C.CourseName, H.TeacherID, U.User-
Name, H.Title,
      H.PublishTime,  IsAnswer=case when len(A.Content)>0 then '已回答' else '未回答'
end ,
      IsReply=case when len(A.TeacherComment)>0 then '已批阅' else '未批阅' end
      FROM    HomeWork H
      LEFT JOIN  Answer  A  ON  H.ID= A.HomeworkID AND A.StudentID=@StudentID
      INNER JOIN  Course C ON C.ID=H.CourseID
      INNER JOIN  [User] U ON U.ID= H.TeacherID
注:len()是 Sql Server 的系统函数,获取字段长短
```

该 SQL 语句比较复杂,IsAnswer 是显示学员是否回答。当 Answer 表中的 Content 字段长度大于 0 时,说明学员已经回答,IsAnswer 为"已回答";反之,则为"未回答"。IsReply 与 IsAnswer 字段类似。HomeWork 表与 Answer 表之间是左连接,因为若为内连

接时，当学员没有回答作业，Answer 表中的 HomeworkID 字段没有当前作业的编号，导致 H. ID=A. HomeworkID 条件不成立，则查询为空。若为左连接，就算学员没有回答，没有当前作业的编号，也不影响整个作业情况的查询结果。而 Course 和 User 表，显然不会出现这种情况，所以为内连接。

当单击第一列中的作业标题(例如“语文教学的改革”)时，则会跳转到 Transfer. Aspx 页面，该页面是一个过渡页，处理一些中间逻辑，代码如程序 5-76 所示。

程序 5-76　Transfer. Aspx. cs 代码

```
protected void Page_Load(object sender, EventArgs e)
        {
            string HomeworkID=Request.QueryString["HomeworkID"].ToString();
            string StudentID=((UserData)Session["Current_User"]).UserID;
            object[] Params=new object[]{HomeworkID,StudentID};
            int mark=IndexObject.GetMark(Params);
            //如果 mark 为 1,则跳转到 ReadResult.aspx,否则说明没有回答作业,跳转到 Add-
Answer.aspx
            if (mark==1){
                    Response.Redirect("ReadResult.aspx?HomeworkID="+HomeworkID+"
&StudentID="+StudentID);
            }
            else{
                    Response.Redirect("AddAnswer.aspx?HomeworkID="+HomeworkID+
"&StudentID="+StudentID);
            }
        }
```

页面加载时，主要执行的是 IndexObject 类的 GetMark 方法，该方法的参数是作业编号(HomeworkID)、学员编号(StudentID)，主要功能是判断学员是否回答目前作业，返回一个标志位。当学员没有回答，则跳转到回答作业页面(AddAnswer. aspx)，否则跳转到查看结果页面(ReadResult. aspx)，并把参数都带过去。所以用户实际上是看不到该页面的。GetMark 方法涉及的 SQL 语句如下所示。

```
SELECT [ID] FROM Answer WHERE HomeworkID=@HomeworkID and StudentID=@StudentID
```

下面先分析一下回答作业页面(AddAnswer. aspx)，如图 5-38 所示。

该页的前台 HTML 比较简单，就不再详细介绍。页面初始化时，会把作业的相关信息显示在页面上，实现的代码如程序 5-77 所示。

程序 5-77　页面初始化代码(AddAnswer. aspx. cs)

```
protected void Page_Load(object sender, EventArgs e)
  {
      if (! IsPostBack)
      {
```

语文教学的改革
发布教师：Teacher1 发布时间：2008-1-2 16:25:10
作业内容：
语文教学的改革非常重要，大家要努力学习语文教学的改革的方法。
回答作业：
确 定　返 回

图 5-38　回答作业页面(AddAnswer. aspx)

```
            string HomeworkID=Request.QueryString["HomeworkID"].ToString();
            object[] Parmas=new object[] { HomeworkID };
            DataTable dt=IndexObject.GetHomeWorkByID(Parmas);
            this.lb_title.Text=dt.Rows[0]["Title"].ToString();
            this.lb_teacher.Text=dt.Rows[0]["UserName"].ToString();
            this.lb_time.Text=dt.Rows[0]["PublishTime"].ToString();
            this.txt_Content.Text=dt.Rows[0]["Content"].ToString();
        }
    }
```

如上面代码所示，第一次加载页面时，首先获取查询字符串 HomeworkID 的值，再以该值为参数，调用 IndexObject 类的 GetHomeWorkByID 方法。该方法的功能是根据作业编号，获取该作业的基本信息，再对页面上的控件赋值，达到初始化页面的目的。GetHomeWorkByID 方法涉及的 SQL 语句如下所示。

```
SELECT H.Title,U.UserName,H.Content,H.PublishTime FROM HomeWork H
    LEFT JOIN [User] U ON H.Teacherid=U.[ID]
    WHERE H.[ID]=@HomeworkID
```

单击“确定”和“返回”按钮，触发 submit_Click 和 Cancel_Click 事件，代码如程序 5-78 所示。

程序 5-78　提交事件和返回事件的代码(AddAnswer. aspx. cs)

```
protected void submit_Click(object sender, ImageClickEventArgs e)
  {
      string HomeworkID=Request.QueryString["HomeworkID"].ToString();
```

```
        string StudentID=Request.QueryString["StudentID"].ToString();
        string Content=this.txt_ReplyContent.Text;
        object[] Params=new object[] { HomeworkID, StudentID, Content };
        int mark=IndexObject.AddAnswer(Params);
        if (mark==1){
            ShowAlert("提交成功!");
            Response.Redirect("Default.aspx");        }
        else{
            ShowAlert("提交失败!");            }
    }
    protected void Cancel_Click(object sender, ImageClickEventArgs e)
    {
        Response.Redirect("Default.aspx");
    }
```

提交事件主要由 IndexObject 类的 AddAnswer 方法实现，该方法的参数为作业编号(HomeworkID)、学员编号(StudentID)和回答内容(Content)。涉及的 SQL 语句如下所示。

```
INSERT INTO Answer (
                HomeworkID,
                StudentID,
                 AnswerTime,
                    Content
                )
     VALUES (
                @HomeworkID,
                @StudentID,
                getdate(),
                @Content
                )
```

返回事件将当前页面跳转到学员作业展示首页(Default.aspx)。

接着分析查看批阅情况页面 ReadResult.aspx，如图 5-39 所示。

该页的前台 HTML 比较简单，就不再详细介绍。页面初始化时，会把作业和答题情况的相关信息显示在页面上，实现的代码如程序 5-79 所示。

程序 5-79　查看批阅情况页面(ReadResult.aspx.cs)代码

```
protected void Page_Load(object sender, EventArgs e)
{
  string HomeworkID=Request.QueryString["HomeworkID"].ToString();
  string StudentID=Request.QueryString["StudentID"].ToString();

  object[] Parmas=new object[] { HomeworkID };
```

语文教学的改革

发布教师：Teacher1 发布时间：2008-1-2 16:25:10

作业内容：

语文教学的改革非常重要，大家要努力学习语文教学的改革的方法。

回答作业：（回答时间：2008-1-16 22:02:41）

恩，非常赞成，将语文教学的改革进行到底！

评语：

回答的很好！

得分：90

返　回

图 5-39　查看批改情况页面(ReadResult. aspx)

```
    DataTable dt=IndexObject.GetHomeWorkByID(Parmas);
    this.lb_title.Text=dt.Rows[0]["Title"].ToString();
    this.lb_teacher.Text=dt.Rows[0]["UserName"].ToString();
    this.lb_time.Text=dt.Rows[0]["PublishTime"].ToString();
    this.txt_Content.Text=dt.Rows[0]["Content"].ToString();

    object[] Parmas1=new object[] { HomeworkID ,StudentID};
    DataTable dt1=IndexObject.GetResult(Parmas1);
    this.txt_ReplyContent.Text=dt1.Rows[0]["Content"].ToString();
    this.lb_AnswerTime.Text=dt1.Rows[0]["AnswerTime"].ToString();
    if (dt1.Rows[0]["TeacherComment"]!=null)
    {
        this.txt_Comment.Text=dt1.Rows[0]["TeacherComment"].ToString();
    }
    if (dt1.Rows[0]["HomeworkS"]!=null)
    {
        this.lb_result.Text=dt1.Rows[0]["HomeworkS"].ToString();
    }
}
```

首先，调用 IndexObject 类的 GetHomeWorkByID 方法，该方法在前面已经讲过。接着调用 IndexObject 类的 GetResult 方法，该方法的参数为作业编号(HomeworkID)、学员编号(StudentID)，功能是获取学员答题和教师批阅的信息。这两个方法调用完后，把获取的信息初始化到页面的控件上。GetResult 方法涉及的 SQL 语句如下所示。

```
SELECT Content,TeacherComment,HomeworkS,AnswerTime FROM Answer
WHERE HomeworkID=@HomeworkID AND StudentID=@StudentID
```

单击“返回”按钮，触发返回事件，跳转到学员作业展示的主页面。

介绍完回答作业页面和查看结果页面后，接下来再把学员作业展示的主页面(Default.aspx)中的查询功能介绍一下。查询条件可为“教师”或“课程”。当选择教师查询时，只能选择教师，如图 5-40 所示；当选择课程查询时，只能选择课程，如图 5-41 所示。实现动态变化的效果，前台 HTML 代码如程序 5-80 所示。

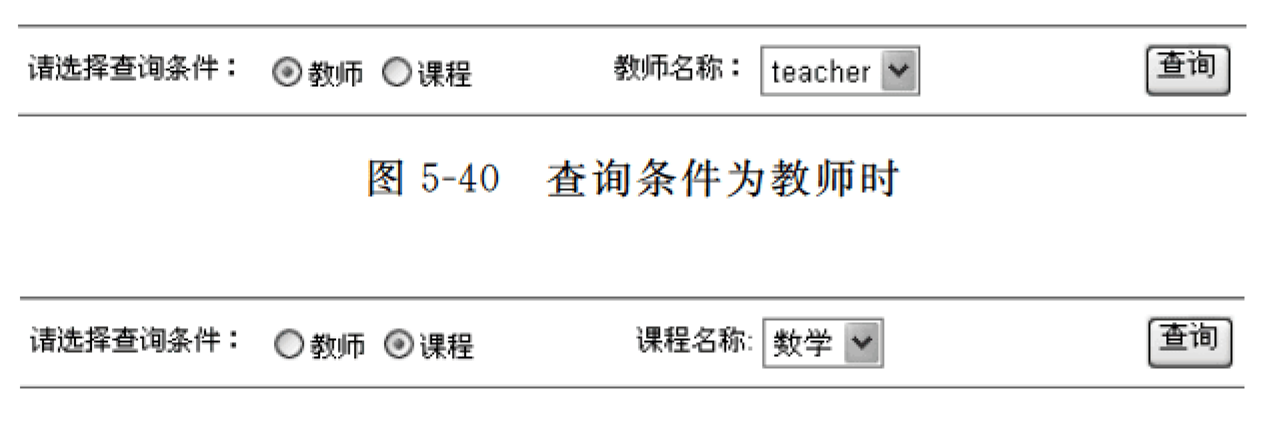


图 5-40　查询条件为教师时

图 5-41　查询条件为课程时

程序 5-80　查询的前台 HTML(Default.aspx)

```
<table>
      <tr>
        <td align="right" width="15%">请选择查询条件：</td>
        <td align="left" style="width: 9%">
            <asp:RadioButtonList ID="rbl_Condition" runat="server" AutoPostBack="
true"
            RepeatDirection="Horizontal" OnSelectedIndexChanged="rbl_Condition_Se-
lectedIndexChanged">
                <asp:ListItem Value="0" Selected="True">教师</asp:ListItem>
                <asp:ListItem Value="1">课程</asp:ListItem>
            </asp:RadioButtonList></td>
          <td align="right"><asp:Label ID="txt_params" runat="server" Text="课程名
称:"></asp:Label>
          </td>
        <td align="left" ><asp:DropDownList ID="ddl_TeaORCourse" runat="server">
                    </asp:DropDownList></td>
          <td align="left"><asp:Button ID="select" runat="server" Text="查询" On-
Click="select_Click" /></td>
      </tr>
   </table>
```

为了实现查询条件动态变化，必须设置 ID 为 rbl_Condition 的 RadioButtonList 的属性 AutoPostBack="True"，使得 rbl_Condition 中选项更改时，能自动提交页面。页面主要有两个事件，RadioButtonList 的更改选定索引触发的事件 rbl_Condition_SelectedIndexChanged，其实现如程序 5-81 所示。

程序 5-81　rbl_Condition_SelectedIndexChanged 的后台代码(Default.aspx.cs)

```
protected void rbl_Condition_SelectedIndexChanged(object sender, EventArgs e)
        {
            int Condition=int.Parse(this.rbl_Condition.SelectedValue);
            ddlDataBind(Condition);
        }
        protected void ddlDataBind(int Condition)
        {
            DataTable dt;
            if (Condition==0)   //表示查询条件是教师
            {
                this.txt_params.Text="教师名称：   ";
                dt=UserManageObject.GetTeachers();
                this.ddl_TeaORCourse.DataSource=dt.DefaultView;
                this.ddl_TeaORCourse.DataTextField="UserName";
                this.ddl_TeaORCourse.DataValueField="ID";
            }
            else        //表示查询条件是课程
            {
                this.txt_params.Text="课程名称：    ";
                object[] Params=new object[] {""};
                dt=CourseManageObject.GetCourses(Params);
                this.ddl_TeaORCourse.DataSource=dt.DefaultView;
                this.ddl_TeaORCourse.DataTextField="CourseName";
                this.ddl_TeaORCourse.DataValueField="ID";
            }
            this.ddl_TeaORCourse.DataBind();
        }
```

当 rbl_Condition 的选项更改时，触发的 rbl_Condition_SelectedIndexChanged 事件的功能如下：若当前 rbl_Condition 的值为 0，说明 Condition 选择的是“教师”，则将 ID 为 txt_params 的 Label 赋值为“教师名称：”，并由 UserManageObject 类的 GetTeachers 获取所有教师的列表，并以该列表为数据源，将 ID 为 ddl_TeaORCourse 下拉框进行数据绑定；当前 rbl_Condition 的值为 1，说明 Condition 选择的是“课程”，则将 ID 为 txt_params 的 Label 赋值为“课程名称：”，并由 CourseManageObject 类的 GetCourses 获取所有课程的列表，并以该列表为数据源，将 ID 为 ddl_TeaORCourse 下拉框进行数据绑定。实现了动态变化查询条件。GetTeachers 和 GetCourses 方法涉及的 SQL 语句前面已经讲过，在这里不再讲述。

单击“查询”按钮触发事件 select_Click，其实现代码如程序 5-82 所示。

程序 5-82　select_Click 的后台代码(位于 Default.aspx.cs 中)

```
        protected void select_Click(object sender, EventArgs e)
        {
            int mark=int.Parse(this. rbl_Condition.SelectedValue.ToString());
            DataTable dt=GetHomeworkList(mark);
            this.GV_HomeWorkList.DataSource=dt.DefaultView;
            this.GV_HomeWorkList.DataBind();
        }
        private DataTable GetHomeworkList(int mark)
        {
            DataTable dt=null;
            string StudentID=((UserData)Session["Current_User"]).UserID;
            object[] Parmas;
            switch (mark)
            {
                case 0:   //教师
                    {
                        Parmas=new object[] { "", this.ddl_TeaORCourse.SelectedValue,
    StudentID };
                        dt=IndexObject.GetHomeWorks(Parmas);
                        break;
                    }
                case 1:  //课程
                    {
                        Parmas=new object[] { this.ddl_TeaORCourse.SelectedValue, "",
    StudentID };
                        dt=IndexObject.GetHomeWorks(Parmas);
                        break;
                    }
            }
            return dt;
        }
```

查询时，首先根据 rbl_Condition 的值判断查询条件是教师还是课程，若是教师，则调用 IndexObject 的 GetHomeWorks 方法、教师姓名(ddl_TeaORCourse. SelectedValue)、学员编号(StudentID)，得到满足条件的作业列表；若为课程，也是调用 IndexObject 的 GetHomeWorks 方法，不过参数为课程名称(ddl_TeaORCourse. SelectedValue)和学员编号(StudentID)，得到满足条件的作业列表。GetHomeWorks 方法涉及的 SQL 语句前面已经讲过，不过是没有条件约束(即 WHERE 语句)，则在刚提到的两个 GetHome-Works 所涉及的 SQL 语句分别比前面讲到的 SQL 多了一个查询条件(即 WHERE 条件)而已。

5.4.11　思考习题模块(教师端、学员端)

该模块与作业展示模块(教师端、学员端)类似,本书不作阐述。

5.4.12　友情链接模块

友情管理模块也是管理员才具有权限操作的模块。具有的操作权限为删除、修改和增加。该模块主要由 LinkManage 文件夹下的链接管理首页(Default. aspx)、增加链接页面(AddLink. aspx)和修改链接页面(UpdateLink. aspx)这三个页面实现。该模块本书不打算详述,因为其前台代码和后台代码与课程管理类似。下面简单介绍一下用到的核心方法及 SQL 语句。友情链接的首页如图 5-42 所示。



图 5-42　友情链接管理的首页(Default. aspx)

列表初始化(即 GridView 的绑定)由 FriendshipLinkManageObject 类的 GetLinks 方法实现,该方法没有参数。涉及的 SQL 语句如下所示。

```
SELECT [ID],LinkTitle,LinkUrl, AddTime FROM FriendshipLink
```

修改事件由 FriendshipLinkManageObject 类的 UpdateLink 方法实现,该方法的参数为链接编号(LinkID)、链接标题(LinkTitle)和链接地址(LinkUrl)。涉及的 SQL 语句如下所示。

```
UPDATE FriendshipLink
         SET  LinkTitle=@LinkTitle ,LinkUrl=@LinkUrl
   WHERE ID=@LinkID
```

删除事件由 FriendshipLinkManageObject 类的 DeleteLink 方法实现,该方法只有一个参数链接编号(LinkID)。涉及的 SQL 语句如下所示。

```
DELETE FROM  FriendshipLink WHERE [ID]=@LinkID
```

新增链接事件由 FriendshipLinkManageObject 类的 AddLink 方法实现,该方法的参数为链接标题(LinkTitle)和链接地址(LinkUrl)。涉及的 SQL 语句如下所示。

```
INSERT INTO FriendshipLink (
                             LinkTitle,
                             LinkUrl,
                             AddTime
                            )
                  VALUES
                            (
                             @LinkTitle,
                             @LinkUrl,
                             getdate()
                            )
注：getdate()为 SQL Server 中的系统函数，得到当前的时间。
```

友情链接管理模块的所有页面都在系统根目录下的 LinkManage 文件夹中。

5.4.13　个人信息设置模块

个人信息设置有些方面类似于用户注册，但是又有不同于用户注册的方面。图 5-43 是设计的个人信息设置的页面，以不同身份的用户进入管理平台以后，都会有个人信息设置的管理模块，这就要对角色加以区分。

图 5-43　个人信息设置

从图 5-43 可以看出，个人信息设置与注册的最大不同点是它把用户的原有信息在初始阶段就显示出来了，要实现这样的效果，其前台代码片段如程序 5-83 所示。

程序 5-83　个人信息设置页面的 HTML 代码(perinformation.aspx)

```
<table style="width:600px;">
  <tr align="left">
      <td style="height: 26px; width: 108px;">
        <asp:Label ID="Label4" runat="server" Text="用户名: "></asp:Label></td>
      <td style="height: 26px; width: 38px;">
          <asp:TextBox ID="TextBox3" runat="server" MaxLength="18" Width="150px"></
asp:TextBox>
      </td>
      <td style="height: 26px; width: 313px;">
          <asp:Label ID="Label33" runat="server" Text=" * " Width="26px" ForeColor="
Red"></asp:Label>
          <asp:Button ID="Button3" runat="server" BorderStyle="None" Text="验证"
                          OnClick="Button3_Click" Width="49px" />
           <asp:Label ID="Label5" runat="server" Width="144px" ForeColor="red">
</asp:Label></td></tr>
<tr align="left">
      <td style="width: 108px">
          <asp:Label ID="Label6" runat="server" Text="密码: "></asp:Label></td>
       <td style="width: 38px">
          <asp:TextBox ID="TextBox4" runat="server" MaxLength="20" Width="150px"></
asp:TextBox>
      </td>
      <td style="width: 313px">
          <asp:Label ID="Label21" runat="server" ForeColor="Red" Text=" * " Width="
26px"></asp:Label>
       <asp:Label ID="Label22" runat="server" Width="147px" ForeColor="Red"></
asp:Label></td>
    </tr>
...
        <asp:ImageButton ID="Button4" runat="server" ImageUrl="../Image/button22.gif"
OnClick="Button4_Click" Width="63px" /></td>
    <td style="width: 38px; height: 21px">
        <asp:ImageButton ID="Button5" runat="server" ImageUrl="../Image/button09.gif"
OnClientClick="return cancleClick()" OnClick="Button5_Click" /></td>
     <td style="width: 313px; height: 21px"></td></tr>
</table>
```

当用户登录以后，个人信息设置页面中会显示用户原有的信息，要实现这样的功能就需要在页面初始化时绑定数据，通过用户登录以后保存在 Session["Current_User"]中的 UserID 来获得用户的基本信息。程序代码如程序 5-84 所示。

程序 5-84　个人信息设置页面初始化的后台代码(perinformation.aspx.cs)

```
protected void Page_Load(object sender, EventArgs e)
  {
    int userid=Convert.ToInt32(((UserData)Session["Current_User"]).UserID);
    Session["userid"]=userid;
    if (! IsPostBack)
      {
          DataTable PerTable=new DataTable();
          PerTable=IndexObject.GetPerInformation(userid);
          string username=PerTable.Rows[0]["UserName"].ToString();
          TextBox3.Text=username;//用户名
          string password=PerTable.Rows[0]["PassWord"].ToString();
          TextBox4.Text=password;//密码
          …
      }
  }
```

通过调用 IndexObject 的方法 GetPerInformation 来显示用户的原有信息。GetPerInformation 方法所涉及的 SQL 语句如下所示。

```
select
    UserName,PassWord,RealName,IDCardNumber,Gender,City,Workplace,TechPost,Duty,Ad-
dress,Telephone,PostNumber,Cellphone,E-mail,QQ,MSN
from [User]
where ID=@userid
```

个人信息更新的代码类似于用户注册,只不过它是一个典型的数据库 Update 操作,通过调用 checklogin 的方法 UpdateUser 来更新用户信息。UpdateUser 方法所涉及的 SQL 语句如下所示。

```
update [User]
set UserName=@UserName,
    [PassWord]=@PassWord,
    RealName=@RealName,
    City=@City,
    Gender=@Gender,
    Workplace=@Workplace,
    TechPost=@TechPost,
    Duty=@Duty,
    Address=@Address,
    Telephone=@Telephone,
    Cellphone=@Cellphone,
    PostNumber=@PostNumber,
    MSN=@MSN,
```

```
    QQ=@QQ,
    E-mail=@E-mail,
    IDCardNumber=@IDCardNumber
where ID=@userid
```

个人信息设置模块的完整源程序在\WEB\IndividualInformation\perinformation.aspx 和\WEB\IndividualInformation\perinformation.aspx.cs 文件中。

第 6 章　Web 服务器和数据库服务器

精心制作好了网站，下一步的工作就是要在一台工作稳定的服务器上把它发布出来。在网络上启用了 Web 服务器后，就可以使用该服务器将精心制作的网页提供给 Internet 上的用户浏览。SQL Server 2000 提供了在服务器的系统上运行的服务器软件和在客户端运行的客户端软件，连接客户和服务器计算机的网络软件则由 Windows 系统提供。SQL Server 2000 数据库系统的服务器运行在 Windows NT/2000 系统上，负责创建、维护表和索引等数据库对象，确保数据的完整性和安全性，能够在出现各种错误时恢复数据。将数据从服务器检索出来后可以生成备份，以便在本地保留，也可以进行操作。SQL Server 2000 建立于 Microsoft Windows NT/2000 的可伸缩性和可管理性之上，提供了功能强大的客户服务器平台。本章介绍在 Windows 2000 操作系统中如何安装和配置 Web 服务器和数据库服务器。

6.1　IIS 配置与管理

随着 Internet 的发展，传统的局域网资源共享方式已不能满足人们对信息的需求，创建 Internet 信息服务器无疑是人们的最佳选择。信息服务器包括 Web、FTP 和 SMTP 虚拟服务器三个方面，不但实现了公司内部网络的 Intranet 信息服务，而且还可以使公司网络连接到 Internet 上，为公司的远程客户或业务伙伴提供信息服务。

6.1.1　Internet Information Server 5.0 简介

IIS 5.0 与 Windows 2000 捆绑在一起，从而集成了 Windows 2000 的一些先进特性。安装 Windows 2000 时可以选择安装该组件，也可以在安装 Windows 2000 之后，通过“添加/删除组件”工具进行安装。与 IIS 4.0 相比，IIS 5.0 的改进在于可靠性、安全性、性能、管理、编程和 Internet 标准支持等方面的增强。例如在性能上，IIS 5.0 增强了对虚拟服务器数量的支持，IIS 4.0 服务器可以支持大约 250 个虚拟 Web 站点，IIS 5.0 则可以支持数千个站点。

IIS 5.0 的可靠性改进表现在：提供了更多的 Web 进程保护功能，如图 6-1 所示。

Web 应用进程和 IIS 核心服务进程(Internet Service)隔离开来,IIS 核心服务进程在单独的内存空间内,Web 应用进程可以在共享地址空间的进程池(Pooled Process)中,也可以单独隔离(Isolated Process)到另一个地址空间,因此能独立停止和重起每个进程,从而提高了 Web 服务器的可靠性和稳定性,但进程隔离也损耗了一部分系统性能。在安全性方面,IIS 5.0 可以使用 Windows 2000 活动目录功能实现用户身份的验证,也可以结合使用证书和活动目录来验证用户。此外,IIS 5.0 也加快了对进程外应用的执行速度及 ADO 数据库访问的效率。

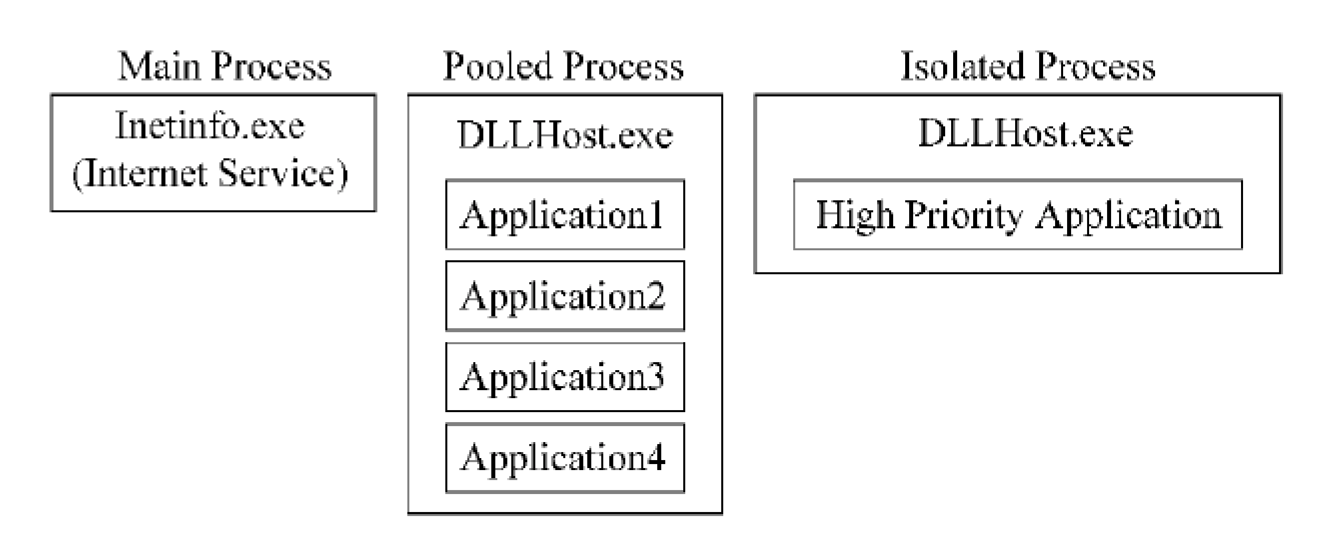


图 6-1　IIS 5.0 的改进结构

在 Windows 2000 高级服务器和数据中心服务器中,IIS 5.0 和群集服务功能紧密集成,通过载量平衡服务可以很方便地建立多机均载的 Web 网站。IIS 5.0 支持动态服务器页面(ASP)的最新版 3.0。与以前的版本相比,ASP 3.0 提高了效率,功能也更强,还提供了和 XML 的集成,同时也可以用 ADSI 2.0 对 Windows 2000 Active Directory 进行操作。

另外,IIS 5.0 还具有以下具体特性。

(1) 摘要式身份验证。允许跨代理服务器和防火墙对用户进行安全和严格的身份验证。另外,还可进行匿名、基本验证及集成 Windows 身份验证,在 Windows NT 下称为质询/响应身份验证和 NTLM 身份验证。

(2) 安全通信。安全套接字协议层(SSL) 3.0 和传输协议层安全(TLS)提供了一种客户端与服务器之间进行信息交换的安全方式。另外,SSL 3.0 和 TLS 还为服务器提供了一种在用户登录服务器之前对客户端进行验证的方法。在 IIS 5.0 中,ISAPI 和 Active Server Pages 都可以访问客户证书,以便编程人员通过其站点跟踪用户。同时,IIS 5.0 还可以将客户证书映射为 Windows 用户账户,以便管理员可以根据客户证书控制对系统资源的访问。

(3) 服务器网关加密。服务器网关加密(SGC)是 SSL 的扩展,允许使用 IIS 的出口版的金融系统采取加密性能更高的 128 位加密。虽然 IIS 5.0 中已内置了 SGC 功能,但使用 SGC 时仍然需要特殊的 SGC 证书。

(4) 安全向导。Web 服务器证书向导、权限向导和 CTL 向导简化了服务器配置的复杂度。

(5) IP 地址及 Internet 域限制。可以授予或拒绝单台计算机、计算机组或者整个域对本机 Web 站点的访问。

(6) 兼容 Kerberos5 身份验证协议。IIS 已完全集成了 Windows 2000 中实现的

Kerberos5 验证协议，从而允许用户在运行 Windows 的计算机之间传递验证凭据。

(7) 证书存储。IIS 证书存储目前已与 Windows CryptoAPI 存储集成在一起。Windows Certificate Manager 提供单一入口，允许用户存储、备份和配置服务器证书。

(8) Fortezza。IIS 5.0 支持称为 Fortezza 的美国政府安全标准。该标准通过一种加密机制保证消息的安全性、完整性、验证，以及对消息、组件和系统的访问控制。

(9) 重新启动 IIS。在 IIS 5.0 中，可以停止并重新启动所有 IIS 管理单元中的 Internet 服务，这使得在应用程序运行不正常或变得不可用时无需重新启动计算机。

(10) 进程账户。进程账户是 IIS 5.0 中的一个新特性，它将字段添加到 W3C 扩充日志文件，以记录关于 Web 点如何使用服务器上 CPU 资源的信息。这些信息用于确定站点是否在使用不相称的过多 CPU 资源或检测故障脚本及 CGI 进程。进程账户仅用于 Web 站点，不提供单个应用程序使用 CPU 的详细细节及仅关于进程外应用程序的日志信息。从进程账户获得的信息可用来决定是否应该启用 Web 站点上的进程限制。

(11) 进程限制。可以限制 CPU 在处理单个 Web 站点进程外的及应用程序的时间百分比。另外，还可以终止和重新启动运行失常的进程。

(12) 改进的自定义错误消息。当 Web 站点出现 HTTP 错误时，目前管理员可以向用户发送消息。可以使用 IIS 5.0 提供的自定义错误消息，也可创建自己的错误消息。

(13) 配置选项。可以在站点、目录或文件级别设置读取、写入、执行、脚本及 FrontPage Web 操作的权限。

(14) 远程管理。IIS 5.0 包含了一些基于 Web 的管理工具，可以从任何平台的几乎所有浏览器上进行 Internet 服务的远程管理。利用 IIS 5.0，可以设置称为"操作员"的管理账户，使之具备一定的 Web 站点管理权限，帮助分担一部分管理任务。

(15) 标准的支持。IIS 5.0 符合 HTTP 1.1 标准，包括 PUT 和 DELETE 等功能及自定义 HTTP 错误消息的能力，并支持自定义的 HTTP 头。

(16) 多个站点，一个 IP 地址。由于支持主机头，因此，可以用一个 IP 地址在运行 Windows 2000 的单台服务器上维护多个 Web 站点。这对于 Internet 服务提供商及维护多个站点的公司非常有用。

(17) Web 分布式创作与版本管理(WebDAV)。允许远程作者通过 HTTP 连接创建、移动或删除服务器上的文件、文件属性、目录和目录属性。在服务器上设置 WebDAV 发布目录与设置虚拟目录一样简单。设置好了发布目录后，具有正确权限的用户就可以将文档发布到服务器上并处理目录中的文件。

(18) PICS 分级。可以将 PICS 分级应用于内容仅适合于成人的站点。

(19) FTP 重新启动。如果在数据传输中出现中断，则可以恢复 FTP 文件下载，而不必再次下载整个文件。

(20) HTTP 压缩。可以更快地在 Web 服务器与启用了压缩的客户之间进行页面传输。压缩和缓存静态文件并对动态生成的文件按需进行压缩。

6.1.2 安装 Internet 信息服务

在安装 Windows 2000 时，如果用户选择了安装 IIS 5.0，系统就会自动创建一个

HTTP 站点和一个 FTP 站点。IIS 预设的 Web 站点和 FTP 站点发布目录也被称为主目录，其 Web 站点主目录的路径是\Inetpub\WWWroot，FTP 站点主目录的路径是\ Inetpub\FTProot。

对于 Web 站点来说，如果本地网络中带有诸如 DNS 这样的命名系统，那么其他访问者只需在浏览器地址下拉列表框中输入计算机名就可以访问相应的站点。但是，如果本地网络中没有诸如 DNS 这样的命名系统，那么其他访问者必须在地址下拉列表框中输入计算机的 IP 地址才能访问。

对于 FTP 站点来说，如果本地网络中带有诸如 DNS 这样的命名系统，那么其他访问者需在地址下拉列表框中输入“FTP：//用户的计算机名”，就可以访问站点。但是，如果本地网络中没有诸如 DNS 这样的一个命名系统，那么其他访问者必须在地址下拉列表框中输入“FTP//用户计算机的 IP 地址”才能访问。

如果用户在安装 Windows 2000 时没有选择安装 IIS 5.0，并需要创建 Internet 信息服务器，则可使用控制面板中的“添加/删除程序”向导来安装此组件，过程如下。

(1) 打开“开始”菜单，选择“设置”→“控制面板”命令，打开“控制面板”窗口，双击“添加/删除程序”图标，打开“添加/删除程序”窗口。

(2) 在左边列表栏中，单击“添加/删除 Windows 组件”按钮，然后单击“组件” 按钮，安装程序开始启动，启动之后打开“Windows 组件向导”对话框(如图 6-2 所示)。



图 6-2　Windows 组件窗口

(3) 选中“组件”列表框中的“Internet 信息服务(IIS)”复选框。

(4) 单击“下一步”按钮，系统即可进行 IIS 5.0 的安装，同时打开“正在配置组件”对话框(如图 6-3 所示)。

(5) IIS 5.0 安装完成之后，向导进入最后一步，单击“确定”按钮即可。

6.1.3　创建 Web 站点

创建 Web 服务器和 FTP 服务器是 Internet 信息服务器最重要的内容，通过 Web 服

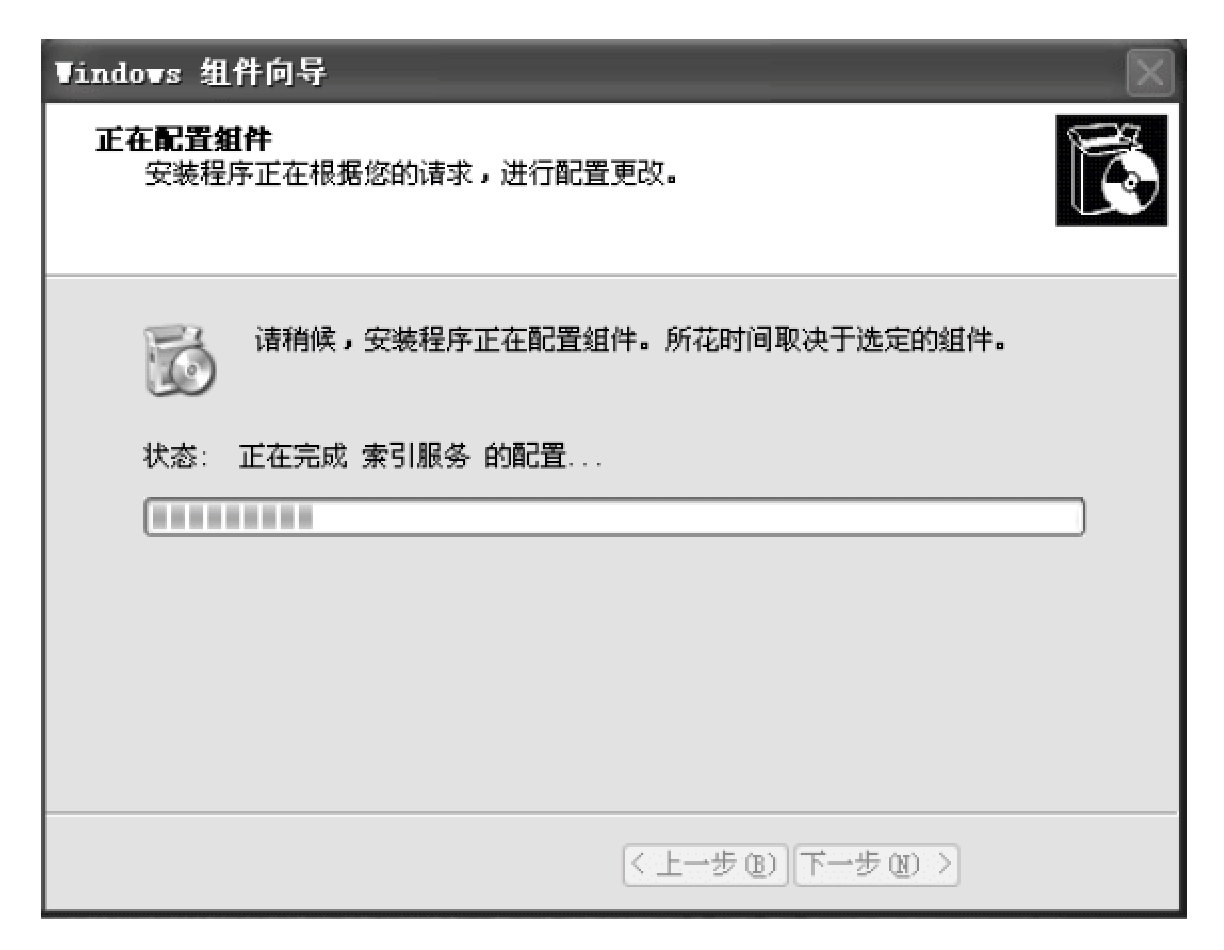


图 6-3 “正在配置组件”对话框

务器，用户可以有效直观地将企业信息发布给企业内部用户和 Internet 远程用户；通过 FTP 服务器，可实现服务器和客户机之间的快速文件传输。安装好 IIS 之后，会自动创建一个默认的 Web 站点和一个默认的 FTP 站点，供用户快速发布内容。用户也可自己创建 Web 站点和 FTP 站点，以扩大和丰富自己的 Web 服务器和 FTP 服务器上的信息。

通常情况下，每一个 Web 或 FTP 站点在内容上都有自己的主题，以便其他访问者快速进行信息查找。如果有多个主题的信息文件需要在网络上发布，应在服务器上创建不同的 Web 或者 FTP 站点，分别进行信息发布。

要创建 Web 或 FTP 站点，可参照下面的步骤。站点创建好之后，就可通过创建主目录和虚拟目录来将具有相关主题的信息文件发布到网上。

(1) 打开 Internet 服务管理器窗口，展开“Internet 信息服务”节点和服务器节点。

(2) 如果要创建 Web 站点，右击服务器节点，从弹出的快捷菜单中选择“新建”→“Web 站点”命令，打开“Web 站点创建向导”对话框。

(3) 单击“下一步”按钮，打开“Web 站点说明”对话框，在“说明”文本框中输入站点说明。

(4) 单击“下一步”按钮，打开“IP 地址和端口设置”对话框，在“输入 Web 站点使用的 IP 地址”下拉列表中选择或直接输入 IP 地址；在“此 Web 站点应使用到的 TCP 端口”文本框中输入 TCP 端口值，默认值为 80。如果有主机标头，可在“此站点的主机标头”文本框中输入标头；默认为没有。

(5) 单击“下一步”按钮，打开“Web 站点主目录”对话框，在“路径” 文本框中输入主目录的路径或单击“浏览”按钮选择路径。

(6) 单击“下一步”按钮，打开“Web 站点访问权限”对话框，在“允许下列权限”选项区域中设置主目录的访问权限。选中“读取”复选框，则只给访问者读取权限；选中“写入”复选框，则访问者有修改权限。一般情况下，应禁用“写入” 复选框，不给访问者授予修改权限。

(7) 单击“下一步”按钮，打开“您已成功完成 Web 站点创建向导”对话框。单击“完成”按钮，完成站点创建。

6.1.4　创建虚拟目录

虚拟目录是指除了主目录以外的其他站点发布目录。在客户浏览器中，虚拟目录就像位于主目录中一样，但它物理上并不包含在主目录中。当站点太复杂，或决定在网页中使用脚本或应用程序时，就需要为要发布的内容创建虚拟目录。

要创建虚拟目录，可参照下面的步骤。

(1) 打开 Internet 服务管理器窗口，在创建好的 Web 站点上单击右键，选择“新建”→“虚拟目录”命令，打开“虚拟目录创建向导”对话框，然后单击“下一步”按钮，打开“虚拟目录别名”对话框。

(2) 在“别名”文本框中输入用于获得此 Web 虚拟目录访问权限的别名。输入别名后，单击“下一步”按钮，打开“Web 站点内容目录”对话框。

(3) 如果用户知道目录路径，可直接在“目录” 文本框中输入目录路径，否则单击“浏览”按钮，打开“浏览文件夹”对话框，选择目录路径。

(4) 单击“下一步”按钮，打开“访问权限”对话框。在“允许下列权限”选项区域中，用户可以为此目录设置访问权限。

(5) 访问权限设置完成后，单击“下一步”按钮，打开“您已成功完成虚拟目录创建向导”对话框。单击“完成”按钮，虚拟目录创建完成。

对于 Web 站点来说，如果需要建立多个虚拟目录，上面的方法就显得不太方便。这时，用户可以直接通过设置文件的 Web 共享属性来快速创建虚拟目录，具体操作步骤如下。

(1) 打开“我的电脑”或“资源管理器”窗口，右击要共享的文件夹，从弹出的快捷菜单中选择“属性”命令，打开文件夹属性对话框，然后选择“Web 共享”选项卡(如图 6-4 所示)。

(2) 选择“共享文件夹”单选按钮，此时会弹出“编辑别名”对话框(如图 6-5 所示)。

(3) 在“别名”文本框中输入该目录的别名。按照默认规定，如果没有更改信息，计算机将指定该目录名为匿名。在“访问权限”选项区域中，通过启用复选框来设置虚拟目录的访问权限，例如选中“脚本资源访问”复选框，则允许访问者访问脚本资源。

(4) 在“应用程序权限”选项区域中，通过选择单选按钮来设置目录中应用程序的权限，例如选择“执行(包括脚本)”单选按钮，则允许访问者执行目录中的应用程序及其脚本。

(5) 设置完毕，单击“确定”按钮保存设置并返回到共享文件夹属性对话框，再单击“确定”按钮即可。



图 6-4 文件夹属性对话框



图 6-5 "编辑别名"对话框

6.2 SQL Server 2000 的配置与管理

6.2.1 设置选项

使用企业管理器的图形工具和代码都可以调整 SQL Server 的选项，但是必须要注意的是，在企业管理器和查询分析器中都是无法设置 SQL Server 的全部选项的。虽然企业管理器具有易于使用的优点，可以使用易于理解的对话框，通过点选的方式选择各种选项，在其引导下一步一步地完成相应的配置任务。但是，与在查询分析器中使用脚本进行配置的方式相比，这种方式缺乏可重复性。

1. 配置服务器

服务器级配置选项控制服务器范围内的设置，如 SQL Server 如何使用硬件、它如何在 Windows 平台下执行多线程任务及一个触发器是否能够触发其他的触发器等。在配置服务器时，必须牢牢记住配置的目标在于：一致性和性能。

在服务器的属性对话框中可以用图形化的方式配置许多服务器选项。要启动服务器的属性对话框，可以在控制台树中，在要配置的服务器上单击鼠标右键，并从弹出的快捷菜单中选择"属性"命令。

在企业管理器中，通过"SQL Server 属性(配置)"对话框的"常规"选项卡可以了解服务器的版本和环境，如图 6-6 所示。

使用代码也可以获取同样的信息。例如，可以通过@@Version 全局变量来获取服务器的版本信息：



图 6-6　企业管理器的服务器属性对话框——“常规”选项卡

```
Select  @@Version
```

对于许多配置属性的更改都必须要等到 SQL Server 重新启动才会生效。基于这个原因，在“SQL Server 属性(配置)”对话框的“服务器设置”选项卡中，通过选择对话框底部的单选按钮，可以分别显示改动后的配置值和当前所采用的运行时值。

在代码中，可以使用 sp_configure 系统存储过程来设置许多的服务器选项。如果在执行它时不指定任何参数，它就会返回服务器的当前设置，如下面的代码所示。

```
EXEC sp_configure
```

使用扩展的存储过程——xp_msver，可以返回其他服务器和环境属性的信息。

```
EXEC xp_msver
```

ServerProperty()系统函数是另一种可以用来获取服务器信息的方法。该函数的优点在于：可以在 Select 语句的表达式中使用它。例如，下面的代码就使用了 ServerProperty()函数来获取 SQL Server 实例的版本信息。

```
Select ServerProperty('Edition')
```

2. 配置数据库

数据库级的选项用于配置当前数据库在 ANSI 兼容性和恢复方面的设置。

在企业管理器的数据库属性对话框中可以设置绝大多数的数据库选项。要打开这个对话框，可以在控制台树中，使用鼠标右键单击要操作的数据库，并从弹出的快捷菜单中选择“属性”命令。该对话框的“选项”选项卡如图 6-7 所示。

在代码中，使用 sp_dboption 系统存储过程可以对数据库配置选项进行设置。如果



图 6-7 使用企业管理器数据库属性对话框中的"选项"选项卡来配置绝大多数常用的数据库属性

在执行它时不指定任何参数,它就会列出所有可用的数据库设置。

```
EXEC sp_dboption
```

3. 配置连接

许多连接级选项都是用于设置 ANSI 兼容性或者连接性能的。

连接级选项的作用域非常有限。如果是在交互式会话中设置了连接级的选项,那么一旦再次设置了该选项的值,或者会话结束了,刚才对于选项的设置就失效了。如果是在存储过程中设置的连接级选项,那么只有在执行包含它的存储过程时,这一设置才是有效的。

连接级选项通常是通过 Set 命令来设置的。下面的代码对在当前会话中 SQL Server 处理空值的方式进行了设置。

```
Set Ansi_nulls off
```

设置连接属性的另一种方法是使用 SessionProperty()函数。

```
Select SessionProperty('ANSI_NULLS')
```

当查询分析器建立到服务器的连接时会设置一些连接属性。要查看连接属性的默认值,可以在查询分析器中选择"工具"→"选项"命令,并在打开的"选项"对话框中选择"连接属性"选项卡。如果要对当前的连接级选项进行设置,则可以使用"查询"→"当前连接属性"命令,如图 6-8 所示。



图 6-8　可以使用查询分析器的“当前连接属性”对话框来查看或者设置当前会话的连接级属性

6.2.2　启动 SQL Server 数据库服务

安装完成 SQL Server 2000 后，在程序菜单里找到 Microsoft SQL Server 2000 程序组，单击“服务管理器”菜单项以启动 Microsoft SQL Server 服务管理器，如图 6-9 所示。



图 6-9　SQL Server 服务管理器

在“服务器”下拉列表中选择一个服务器。如没有启动 SQL Server 服务时，可以单击“开始/继续”按钮以启动 SQL Server 服务。

启动之后，可以单击“暂停”或“停止”按钮来暂停和中断服务。“暂停”按钮用于暂时停止服务，所有设置不改变，重新开始服务时，速度较快；“停止”按钮用于关闭该服务，所有设置被储存，再次开始服务时，重新调入数据及设置，速度较慢。

6.2.3　使用 SQL Server 企业管理器

SQL Server 企业管理器如图 6-10 所示。

在 SQL Server 组中的 NIUJIYIN-9EAECO(Windows NT)服务器中，有几个文件夹用于管理 SQL Server 数据库。

1. 数据库

在数据库列表中列出了这个数据源中包含的数据库。有一些是系统自带的数据库，如 master、model、msdb、Northwind、pubs 和 tempdb；有一些是用户自己建立的数据库，

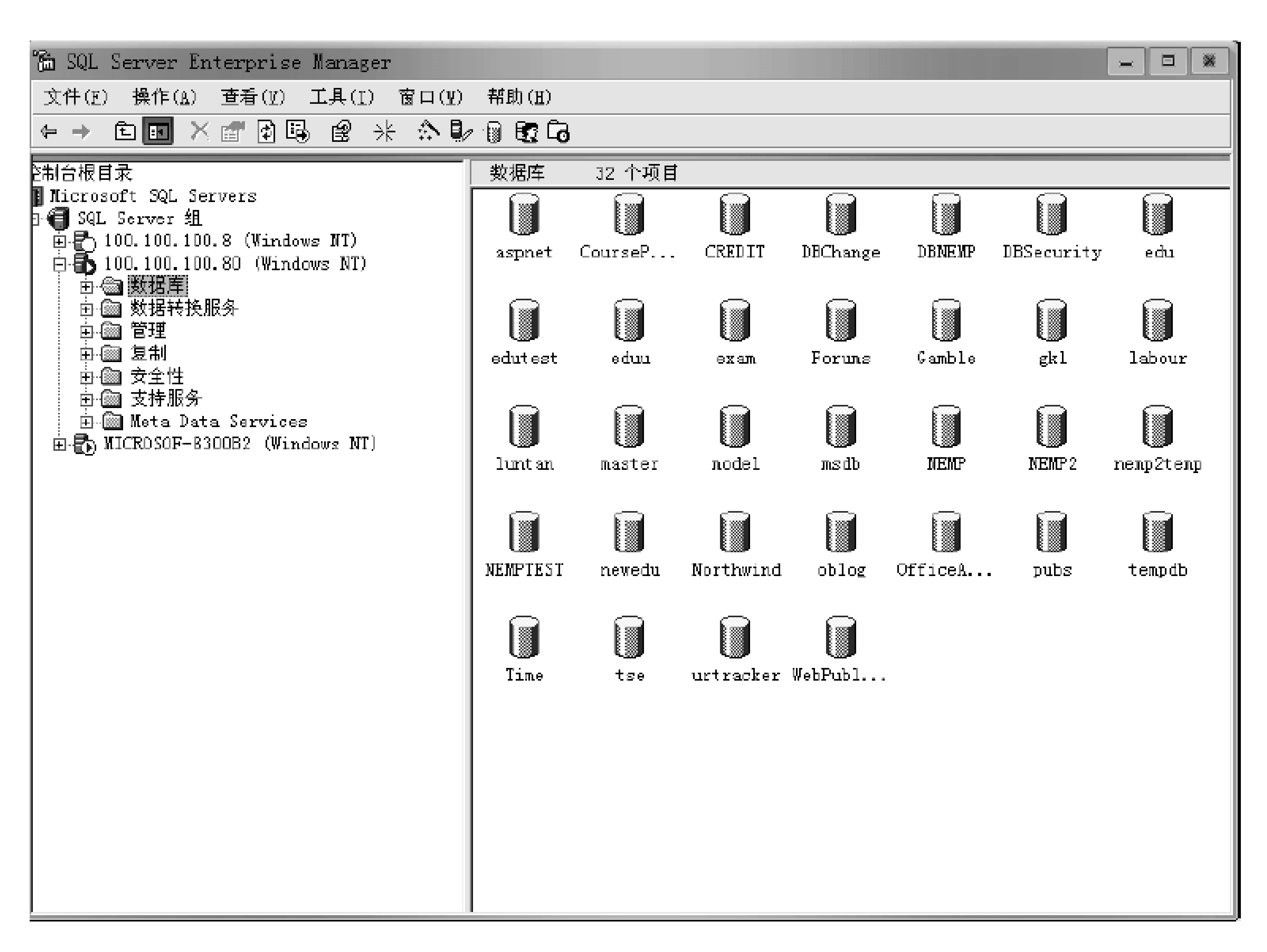


图 6-10 SQL Server 企业管理器

如 DBNEMP（如图 6-11 所示）和 newedu。

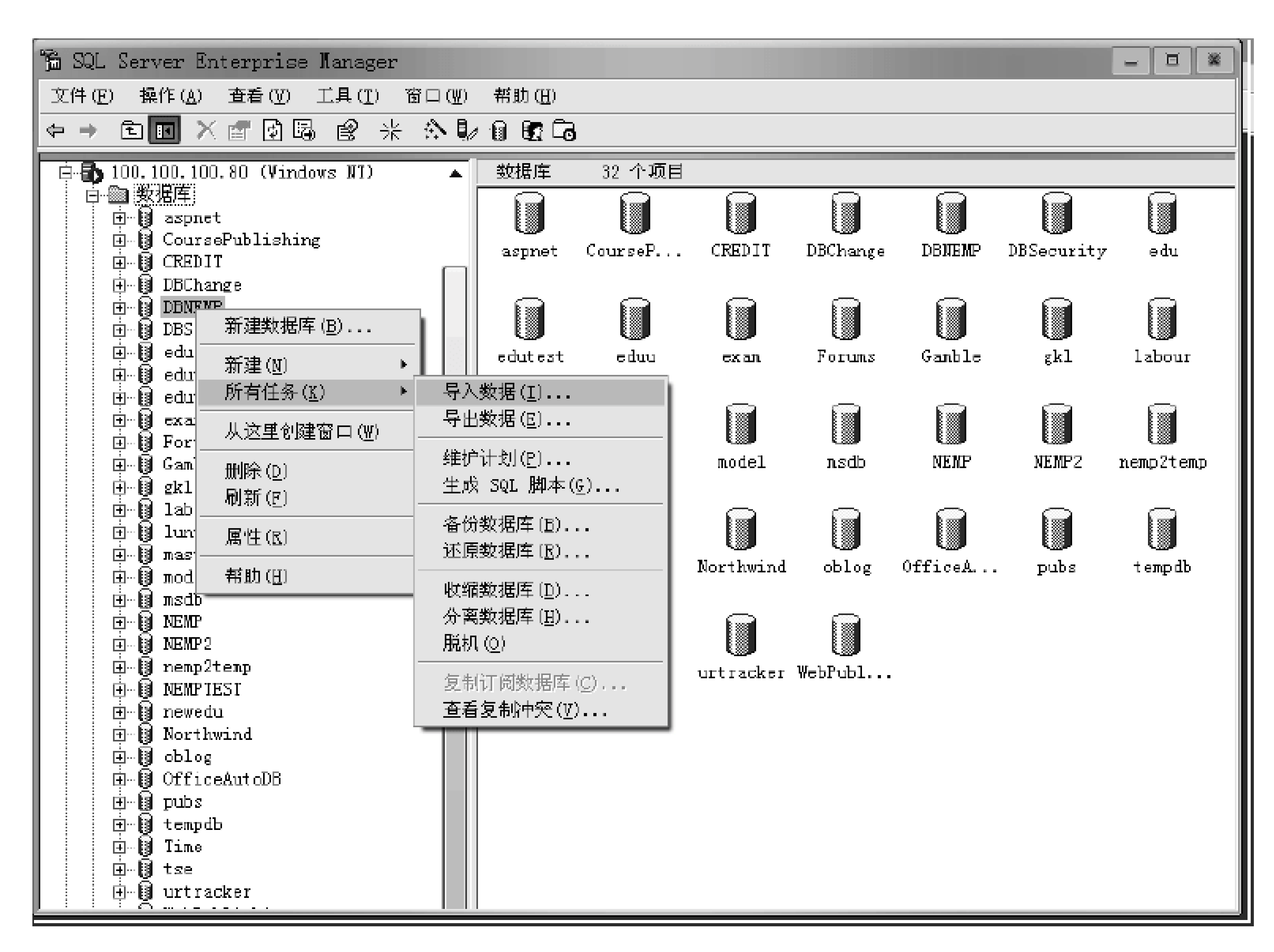


图 6-11 DBNEMP 数据库

可以用右键单击左边树形控件里的 NEMP2 实现对数据库的管理，管理的内容包括设置数据库属性、视图、导入数据、导出数据、备份数据库、还原数据库及生成 SQL 脚本等，如图 6-12 所示。

（1）展开 DBNEMP 下的“表”节点，如图 6-12 所示。

图 6-12　DBNEMP 数据库中的表

在右边信息窗口中会列出该数据库中所有的表，包括建立数据库时 SQL Server 自动建立的和用户建立的。

从返回信息中可以看出表的名称、所有者、类型和创建日期，单击相应按钮可以按该属性排序。在这些表中，System 类型的表很多，它们是 SQL Server 用于维护该数据库而自动创建的，初学者不应该擅自更改这些表，否则有可能导致所建立的数据库无法正常运行，从而导致整个网站瘫痪。

用户可以右键单击空白处，在弹出的快捷菜单中选择“新建表”命令或者右键单击树中的“表”节点，在弹出的快捷菜单中选择“新建表”命令，这两种方法都可以创建一个新表。

另外，通过单击右键，在弹出的快捷菜单中还可以选择完成以下功能。

- 设计表。
- 查询表。
- 设置访问权限。

（2）展开 DBNEMP 下的“用户”节点，可以查看和设置该数据库的用户。

右边信息窗口中列出了有该数据库访问权限的用户，如图 6-13 所示。可以右键单击空白区域，在弹出的快捷菜单中选择“新建数据库用户”命令增加该数据库的用户，或者右键单击某一个用户，在弹出的快捷菜单中选择“属性”命令来更改该用户的权限属性。

（3）展开 DBNEMP 下的“角色”节点，可以查看和设置该数据库中的角色。

右边信息窗口中列出了该数据库中的所有角色，如图 6-14 所示。可以右键单击空白区域，在弹出的快捷菜单中选择“新建数据库角色”命令增加该数据库的角色，或者右键单

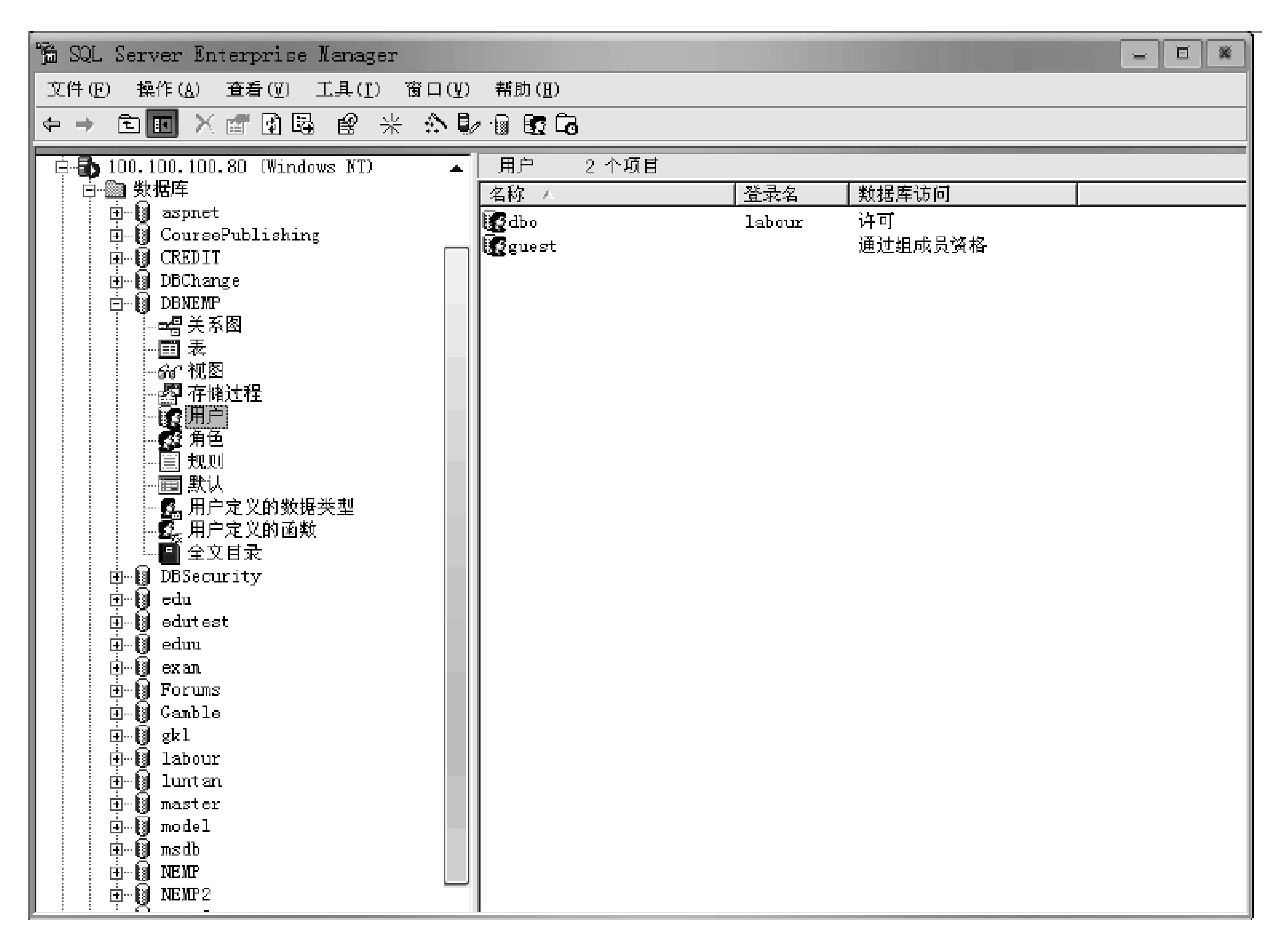


图 6-13 DBNEMP 数据库中的用户

击某一个角色，在弹出的快捷菜单中选择“属性”命令来更改该角色的权限属性及包含的用户。

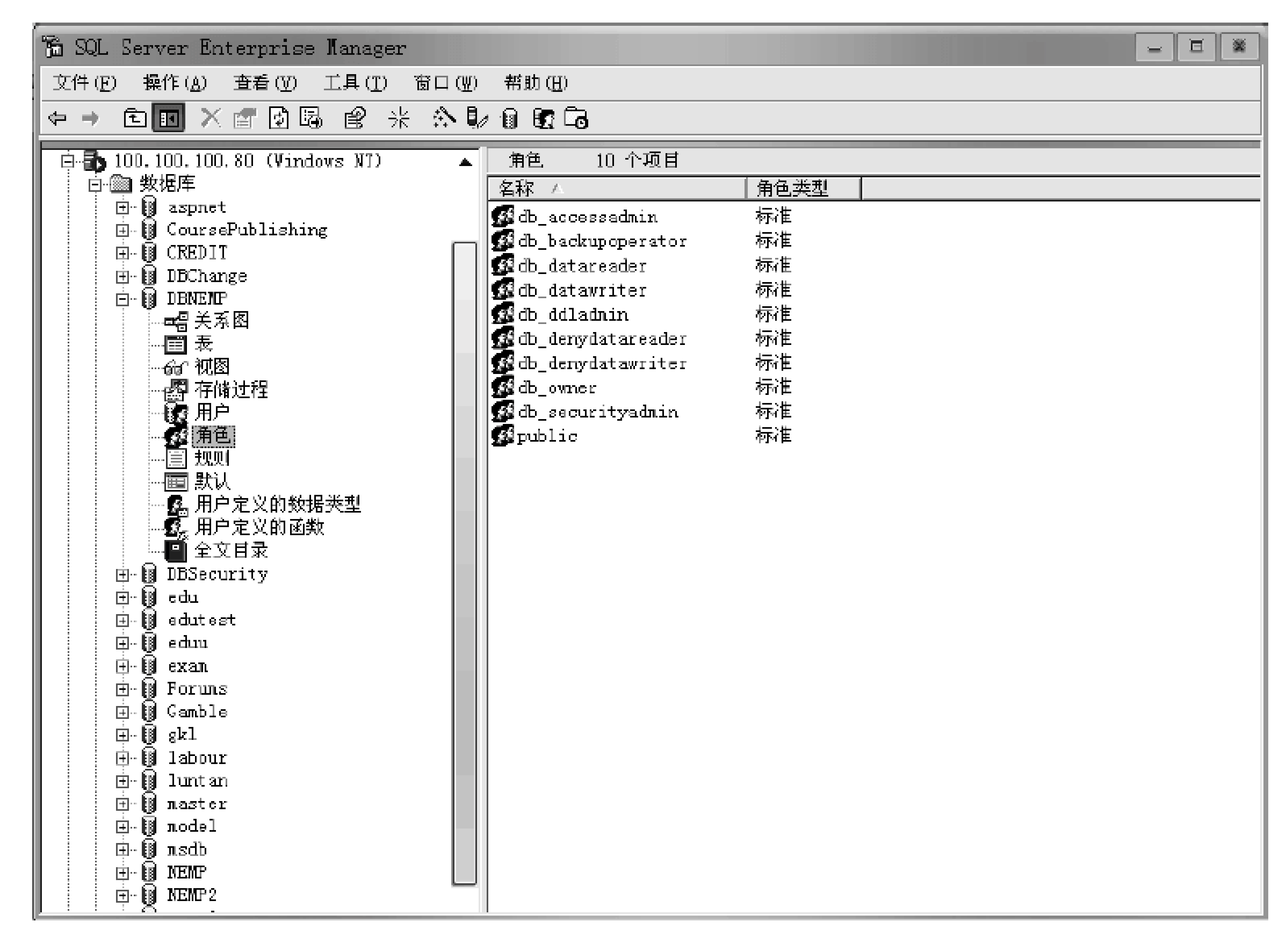


图 6-14 DBNEMP 数据库中的角色

2. 数据转换服务

数据转换服务(Digital Theatre System，DTS)允许在 Microsoft SQL Server、OLE

DB、ODBC 或文本文件(*.TXT)等结构的多个数据源之间输入和输出数据,以及在运行 SQL Server 2000 的多台计算机之间执行数据库和数据库对象的传输。程序员还可以用 DTS 来执行数据转换,以便用它来通过一个在线事务处理系统建立数据仓库和数据商场。DTS 也可以通过批通信程序(Baird Capital Partners,BCP)来输出和输入数据。

数据转换服务设计器(Designer)是一种非常简单快捷的数据导入、导出和转换工具。DTS 设计器可以利用面向事务的工作流引擎设计执行复杂工作流的 DTS 包,进行集成化的数据移动。DTS 设计器使用百分之百的 OLE DB 体系结构,能在众多流行的客户机/服务器数据源间完成复制、转换数据等操作。它提供了与 Access、Excel、FoxPro、SQL Server、dBase Paradox、Oracle、DB-2 和 AS/400 多种数据源的连接,如图 6-15 所示。

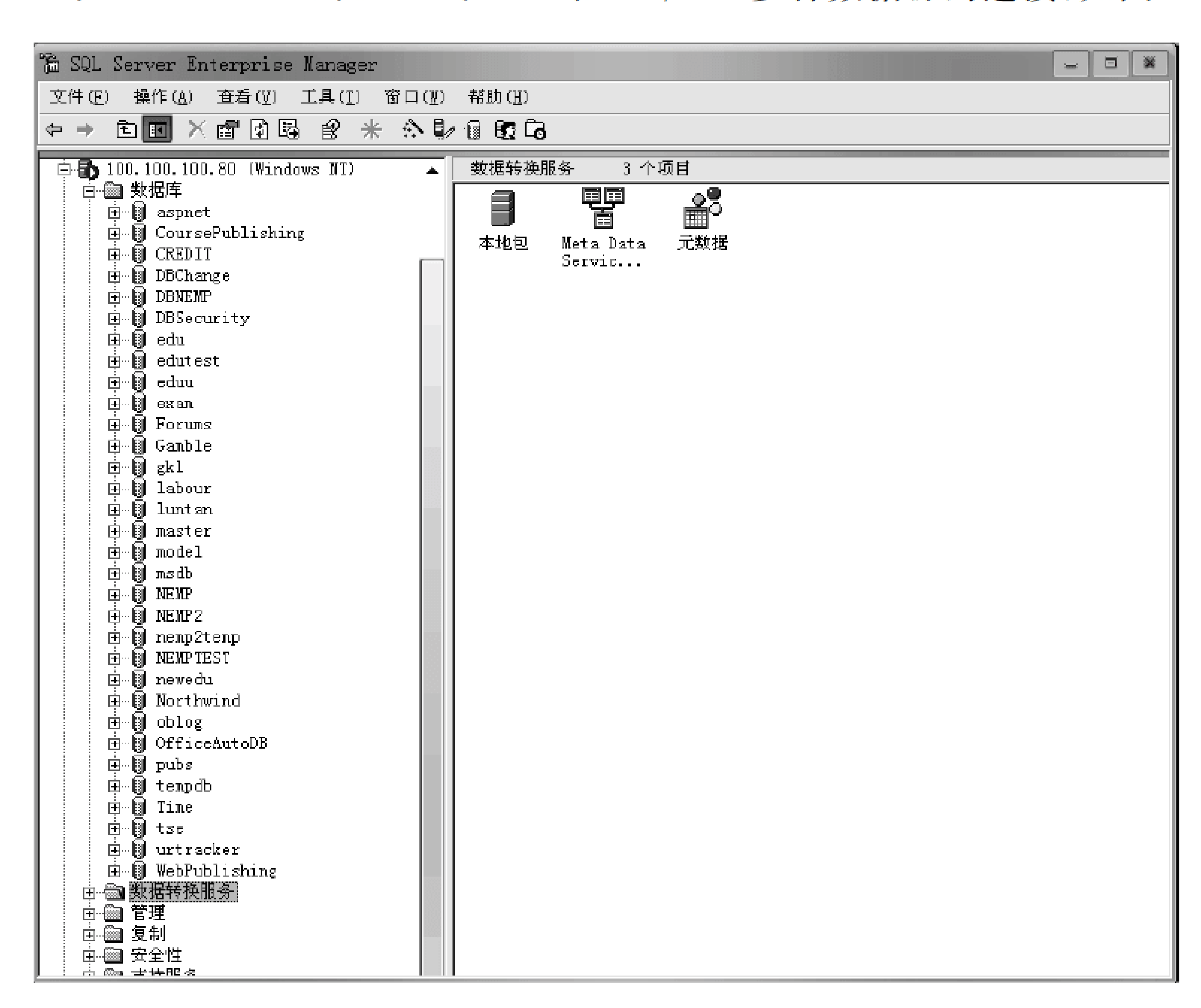


图 6-15　数据转换服务

3. 管理

在这里有着丰富的管理工具,包括 SQL Server 代理、备份、当前活动、数据库维护计划和 SQL Server 日志等,如图 6-16 所示。

其中,备份可以为一个现存的数据库创建一个备份。在电子商务网站中,数据的安全性很重要。虽然数据丢失不可避免,但如果丢失了数据,而没有备份,整个网站就一无所有,不能继续运行了。

当前活动可以实时监控当前数据库状态,如图 6-17 所示。

4. 安全性

在这里用户可以管理数据库服务器的安全属性,如图 6-18 所示。

安全性包括登录、服务器角色、链接服务器和远程服务器几项,它们的功能如下。

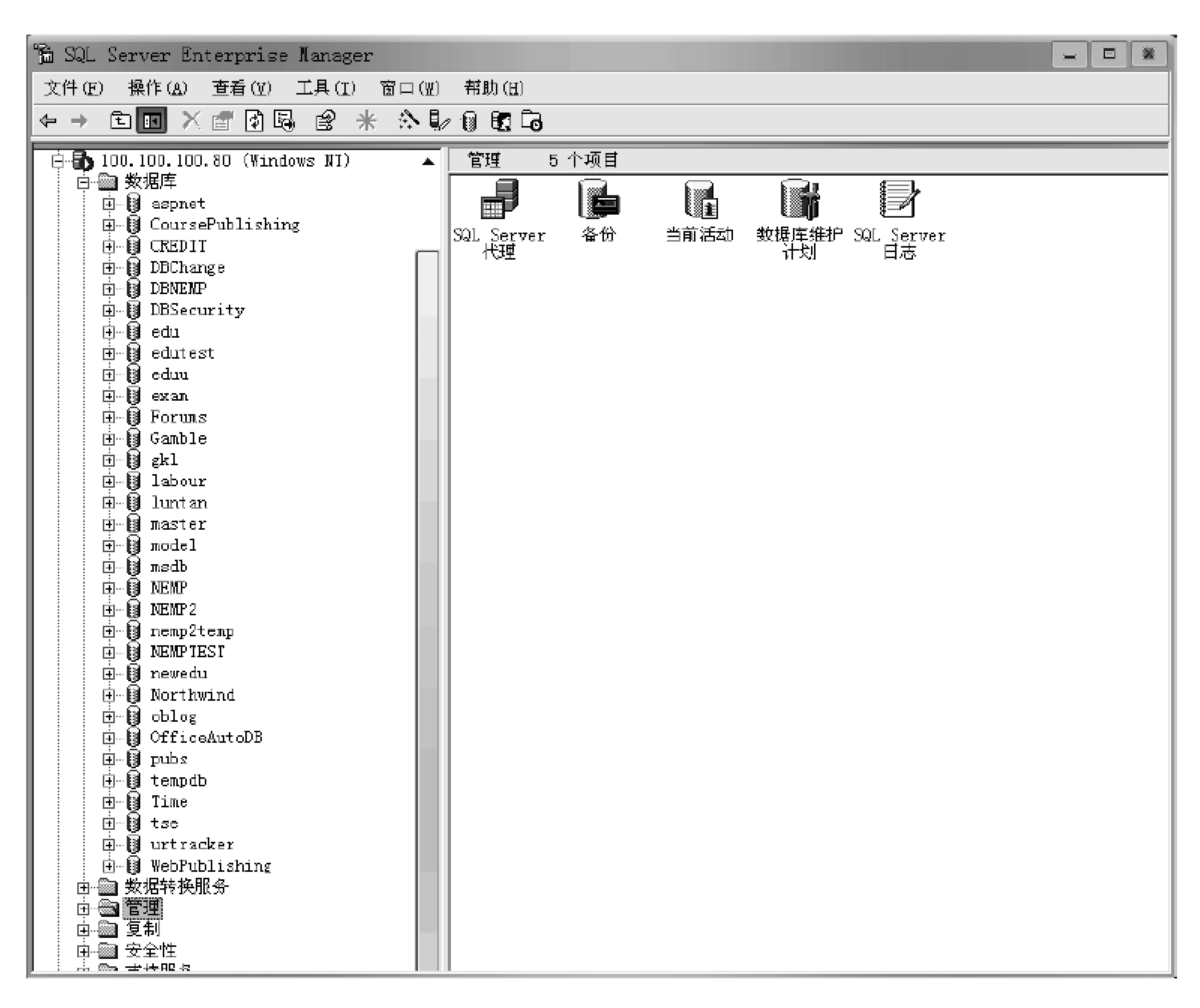


图 6-16 管理

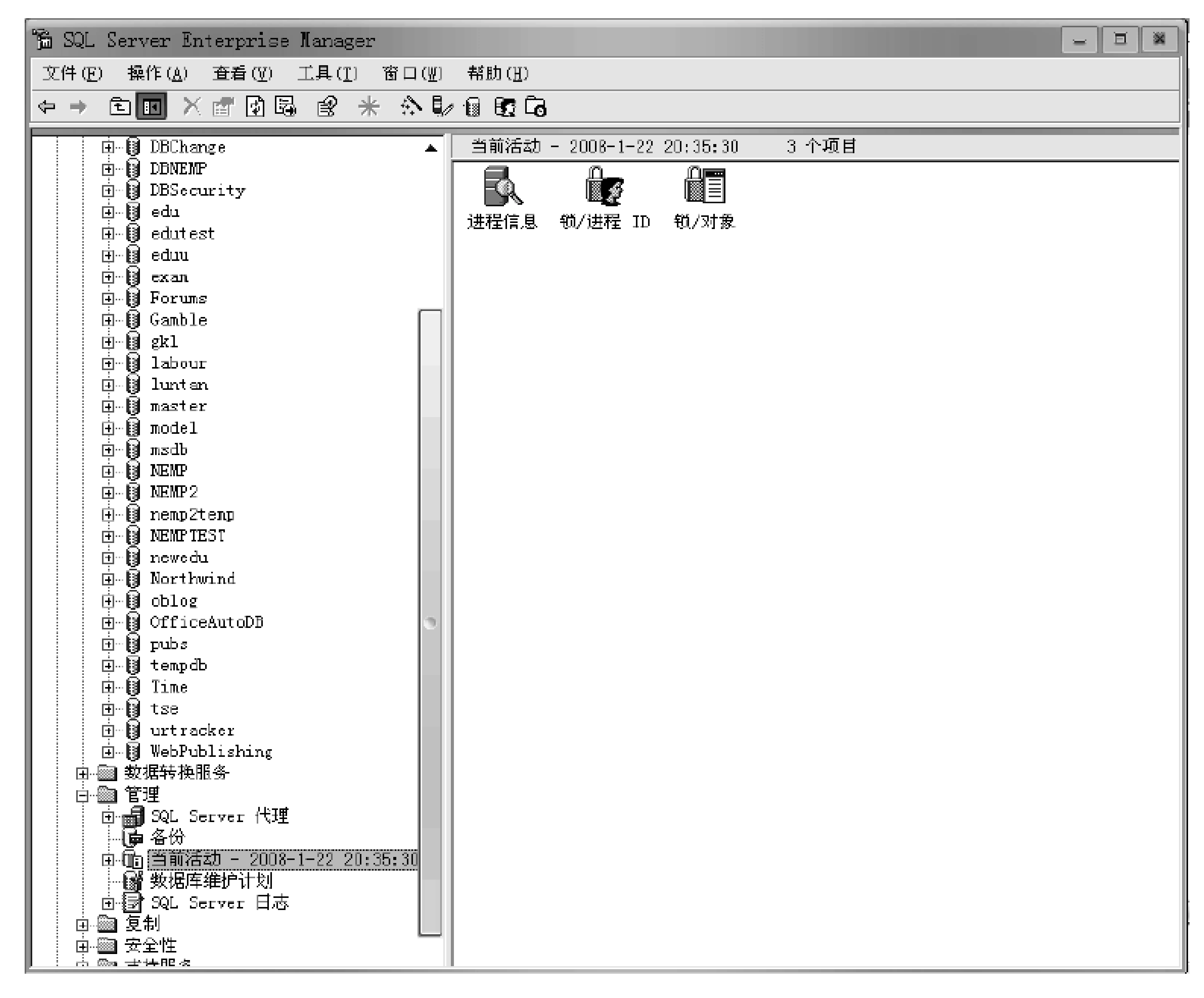


图 6-17 当前活动

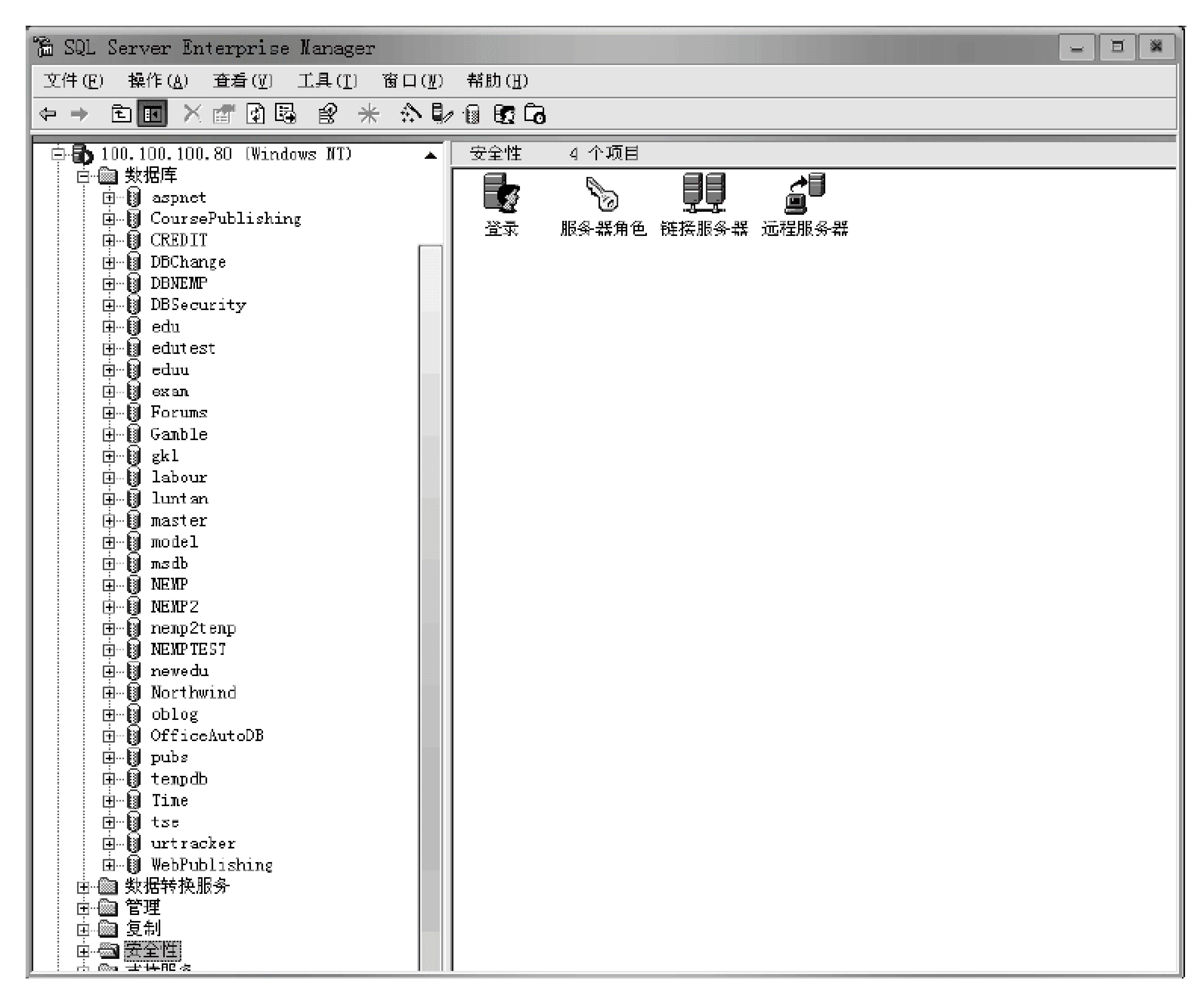


图 6-18　安全性

(1) 登录用于管理使用该 SQL Server 的登录用户。

(2) 服务器角色用于管理该 SQL Server 使用者的角色。

(3) 链接服务器用于管理链接的服务器。

(4) 远程服务器用于管理远程服务器。

提示：这里的 SQL Server 使用者角色与上文提到的数据库的角色不同。前者是针对整个 SQL Server 的使用来说的，它的权限设定关系到整个数据库服务器；而后者是针对这个 SQL Server 服务器中的一个数据库说的，它的权限设置只决定该用户对该数据库的权限，与同一数据库服务器中的其他数据库没有关系。而登录用户即为数据库用户，数据库用户必须先行登录。

展开"登录"节点，在右边信息窗口中列出了该 SQL Server 服务器中所有的注册用户，如图 6-19 所示。

其中有一个内置用户，即 BUIL TIN\Administrators，该用户是 NT Group 组策略认证方式。

另外还有一个默认用户 sa，该用户在安装 SQL Server 时被注册。

其他的用户都是用户自己定义的，除了 BUIL TIN/Administrator 这个用户。

右键单击空白区域，在弹出的快捷菜单中选择"新建登录"命令增加该数据库服务器的注册用户，或者右键单击某一个用户，在弹出的快捷菜单中选择 Properties 命令来更改该用户的权限属性。

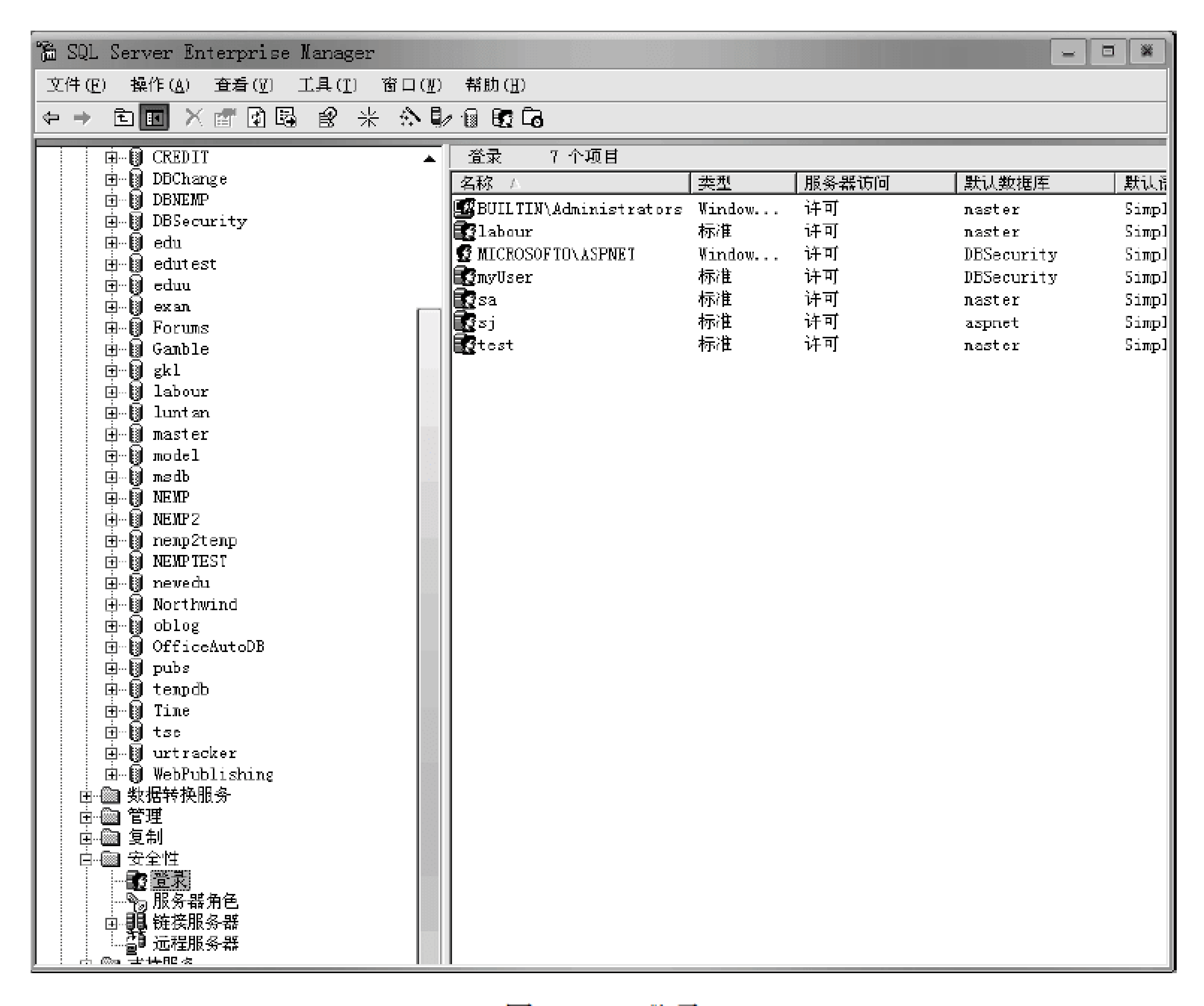


图 6-19 登录

6.2.4 SQL Server 2000 的备份方法

1. SQL Server 2000 备份数据库的 4 种方法

SQL Server 2000 备份数据库的 4 种方法如下。

(1) 数据库备份,即制作数据库中所有内容的副本。在备份过程中,它需要花费的时间和空间最多,不宜频繁进行。恢复时,仅需要恢复最后一次全库备份。该备份以后的修改都将丢失。

(2) 差异备份,即只备份最后一次全库备份后被修改的数据页。备份所耗时间和空间较少。恢复时,先恢复最后一次全库备份,再恢复最后一次差异备份。

(3) 事务日志备份,即只备份最后一次日志备份后所有的事务日志记录。备份所用的时间和空间更少。利用日志备份进行恢复时,可以指定恢复到某一个事务。例如,可以将其恢复到某个破坏性操作执行前一个事务,这是全库备份和差异备份所不能做到的。但利用日志备份进行恢复时,需要重新执行日志记录中的修改命令来恢复数据库中的数据,所以通常恢复的时间较长。建议每周进行一次全库备份,每天进行一次差异备份,每小时执行一次日志备份,这样最多只会丢失一小时的数据。恢复时,先恢复最后一次全库备份,再恢复最后一次差异备份,再依次恢复最后一次差异备份以后进行的所有事务日志备份。

(4) 文件或文件组备份,即备份某个数据库文件或数据库文件组。这种方法必须与日志备份结合才有意义。例如,某数据库中有两个数据文件,一次仅备份一个文件,而且

在每个数据文件备份后，都要进行日志备份。在恢复时，使用事务日志所使用的数据文件恢复到同一个时间点。

备份数据库可以使用备份向导或企业管理器来完成，也可以通过 Transact-SQL 语句来实现。

2. 使用备份向导备份数据库示例

下面以继教网数据库 DBNEMP 为例，介绍使用备份向导备份数据库的方法和步骤。

(1) 在企业管理器中展开服务器组，然后展开一个服务器。

(2) 展开“数据库”文件夹，然后单击要备份的数据库，如 DBNEMP。

(3) 选择“工具”→“向导”命令，然后打开如图 6-20 所示的“选择向导”对话框。

(4) 单击“管理”节点，选择“备份向导”选项，确定后弹出“创建数据库备份向导”对话框，然后单击“下一步”按钮，出现如图 6-21 所示的对话框。

(5) 选择所备份的数据库(如 DBNEMP)，单击“下一步”按钮，出现“输入备份的名称和描述”对话框，在该对话框中输入备份的名称和描述信息，然后单击“下一步”按钮，出现如图 6-22 所示的对话框。

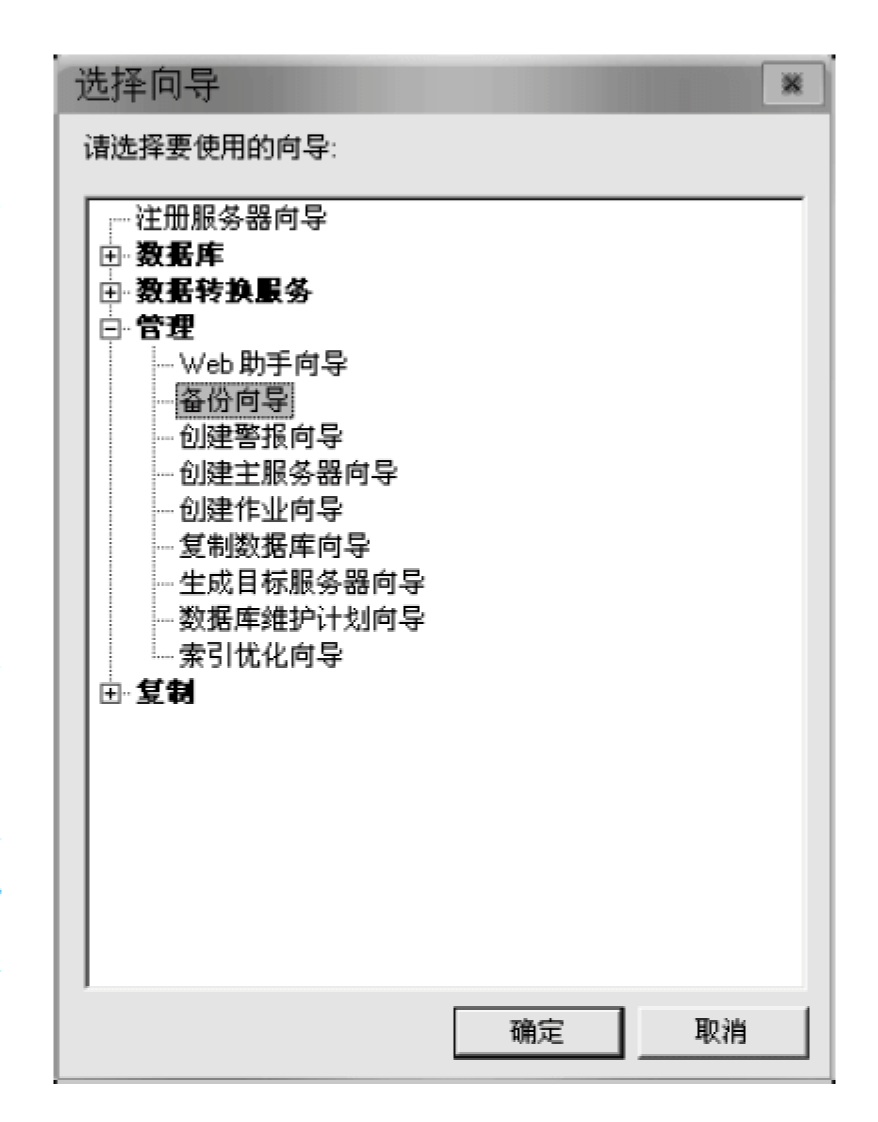


图 6-20　“选择向导”对话框



(6) 选择下列备份方法之一。



图 6-21　选择要备份的数据库

①“数据库备份”：对整个数据库进行备份。

②“差异数据库”：对新的或更改的数据库进行备份。

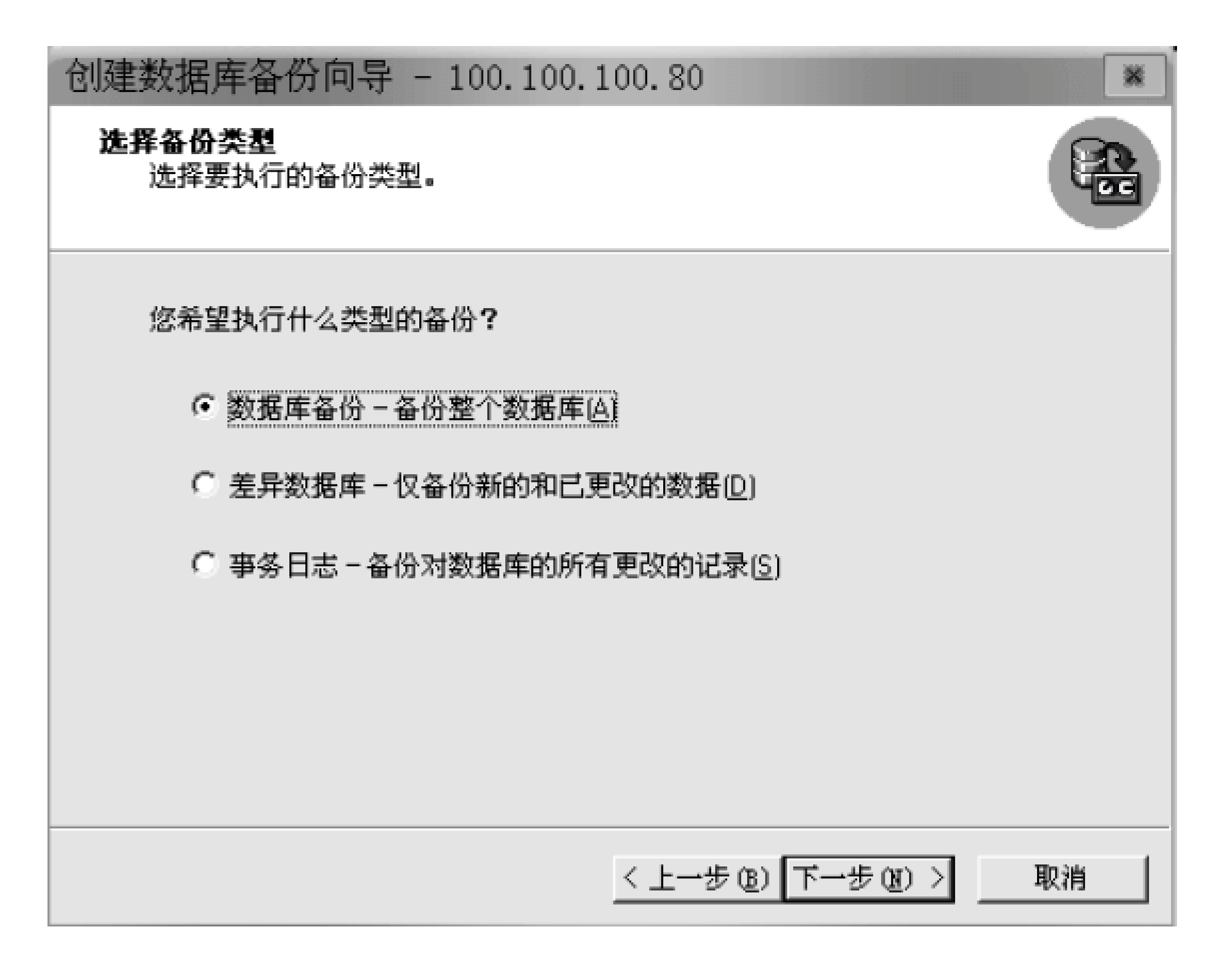


图 6-22 选择备份类型

③“事务日志”:对数据库的所有更改的记录进行备份。

选择一种备份方法之后,鼠标左键单击“下一步”按钮,出现如图 6-23 所示的对话框。



图 6-23 选择备份目的和操作

(7) 选择备份目的和操作。“选择备份设备”选项为“文件”方式,其中文件名和路径自定义。如果要将此次备份追加到原有备份数据的后面,可以选择“属性”选项区域中的“追加到备份媒体”单选按钮;如果要用此次备份的数据覆盖原有备份数据,可以选择“重写备份媒体”单选按钮。单击“下一步”按钮,出现如图 6-24 所示的对话框。

(8) 单击“更改”按钮,确定备份的计划,然后单击“下一步”按钮,出现备份向导的“完成”对话框,如图 6-25 所示。在该对话框中显示刚才所设置的各属性,单击“完成”按钮,如图 6-26 所示。则出现备份进度的对话框,结束后弹出完成对话框,如图 6-27 所示。这样,用向导完成了数据库的备份,并在相应的文件夹内产生了一个.bak 备份文件。



图 6-24　备份验证和调度



图 6-25　完成对话框



图 6-26　“备份进度”对话框



图 6-27　完成对话框

6.2.5 临时、永久备份文件的建立

1. 永久备份文件的建立

执行备份的第一步是创建要包含备份内容的备份文件。在备份执行之前所创建的备份文件称为永久性的备份文件,这些永久性的备份文件也称为备份设备。如果希望重新使用已创建的备份文件,或者设置系统自动备份数据库,就必须使用永久性的备份文件。创建永久性备份文件有两种方法:使用 SQL Server Enterprise Manager,或者执行系统存储过程 sp_addumpdevice。

执行系统存储过程 sp_addumpdevice,可以在磁盘或磁带上创建永久性的备份文件,或者将数据定向到某个命名管道(Named Pipe)中。创建永久性的备份文件时,应该考虑下列因素。

(1) SQL Server 系统在 master 数据库中的系统表 sysdevices 中,创建该永久性备份文件的逻辑名称和物理名称。

(2) 必须指定该备份文件的逻辑名称和物理名称。

(3) 一个数据库最多可以创建 32 个备份文件。

使用 SQL Server Enterprise Manager 创建一个新的永久性备份文件,实际上就是 SQL Server 系统执行系统存储过程 sp_addumpdevice。系统存储过程 sp_addumpdevice 的语法形式如下:

```
[EXECUTE] sp_addumpdevice '设备类型','逻辑名称','物理名称'[,{{controller_type|
'device_status'}}]
```

其中,设备类型是{DISK|TAPE|PIPE},即磁盘、磁带或命名管道。

```
USE master
EXEC sp_addumpdevice 'disk','archivebackupfile','C:\Program Files\Microsoft SQL Server\
MSSQL\BACKUP\arbackupfile.bak'
```

2. 临时备份文件的建立

如果不打算重新使用某些备份文件,就可以创建临时的备份文件。创建临时备份文件有两种方法:使用 SQL Server Enterprise Manager,或者使用 BACKUP DATABASE 语句。SQL Server 系统可创建临时文件,以存储备份操作的结果。在创建临时文件时,必须指定介质类型(磁盘、磁带或命名管道)和完整的路径名称、文件名称。BACKUP DATABASE 语句的语法形式如下。

```
BACKUP  DATABASE {database_name|@database_name_var}
TO<backup_file>[,…,n]
```

其中,<backup_file>是

```
{ backup_file_name|@|backup_file_name_evar}|{DISK|TAPE|PIPE}=
{temp_file_name|@temp_ name_ evar}
```

在磁盘上创建了一个临时备份文件，并且把 DBNEMP 数据库备份到该临时备份文件上。

创建一个临时备份文件，程序清单如下。

```
USE master
BACKUP  DATABASE archive TO DISK='C:\Temp\ TempArchive.bak'
```

使用 BACKUP 语句可以对数据库进行全库备份、差异备份、日志备份或文件和文件组备份。

1）全库备份

全库备份即制作数据库中所有内容的一个副本。从一个全库备份中就可以恢复整个数据库。其语法形式为：

```
BACKUP  DATABASE  数据库名  TO 备份设备名
[WITH [NAME='备份的名称'][,INIT|NOINIT]]
```

在上述语法格式中，备份设备名采用“备份设备类型＝设备名称”的形式；INIT 参数表示新备份的数据覆盖当前备份上的每一项内容，即原来在此设备上的数据都将不存在了；NOINIT 参数表示新备份的数据添加到备份设备上已有内容的后面。

例如，对数据库 DBNEMP 做一次全库备份，并保存在永久备份文件——本地磁盘设备 archivebackupfile 中，而且此次备份覆盖以前所有的备份。代码如下：

```
BACKUP  DATABASE archive TO DISK='archivebackupfile'
WITH INIT, NAME='archivebackup'
```

2）差异备份

差异备份即从最近一次全库备份结束以来，所有改变的数据的备份。当数据库从上次备份以来，数据发生很少的变化时，适合使用差异备份。其语法格式如下：

```
BACKUP  DATABASE  数据库名  TO 备份设备名
WITH DIFFERENTIAL[, NAME='备份的名称'] [,INIT|NOINIT]]
```

其中，DIFFERENTIAL 子句的作用是，指定只对在创建最新的数据库备份后，数据库中发生变化的部分进行备份。

对数据库 DBNEMP 进行差异备份，并保存在永久备份文件——本地磁盘设备 archivebackupfile 中，代码如下：

```
BACKUP  DATABASE archive TO DISK='archivebackupfile'
WITH DIFFERENTIAL, NAME='archivebackup', NOINIT
```

3）日志备份

日志备份即从最后一次日志备份以来所有事务日志备份到备份设备。通常，日志备份经常与全库备份和差异备份结合使用，其语法格式如下：

```
BACKUP  LOG  数据库名  TO 备份设备名
[WITH [NAME='备份的名称'][,INIT|NOINIT]]
```

对数据库 DBNEMP 进行日志备份，并保存在永久备份文件——本地磁盘设备 archivebackupfile 中，代码如下：

```
BACKUP  LOG  archive TO DISK='archivebackupfile'
WITH NAME='archivebackup', NOINIT
```

4）文件和文件组备份

当一个数据库很大时，对整个数据库进行备份可能花费很多时间，这时可以采用文件和文件组备份方式，即对数据库中的部分文件或文件组进行备份。其语法格式如下：

```
BACKUP  DATABASE 数据库名
FILE='文件的逻辑名称'|FILEGROUP='文件组的逻辑名称' TO 备份设备名
[WITH  NAME='备份的名称'] [,INIT|NOINIT]
```

使用上述语法格式备份数据库时，如果备份是文件，则写做“FILE='文件的逻辑名称'”的方式；如果备份是文件组，则写做“FILEGROUP='文件组的逻辑名称'”的方式。

将 DBNEMP 数据库的 archive_data 文件备份到本地磁盘设备 filebackup。代码如下：

```
BACKUP  DATABASE archive FILE='archive_data' to DISK='filebackup'
```

6.2.6 继教网数据库系统的设计

前面用了大量篇幅对 SQL Server 2000 数据库的使用进行了介绍，希望对读者后面的学习有所帮助。

1. 建立数据库 DBNEMP

下面介绍如何为继教网建立一个数据库 DBNEMP。建立该数据库的步骤如下。

（1）打开 SQL Server 企业管理器。

（2）选择 Microsoft SQL Server→SQL Server 组→100.100.100.80(Windows NT)→数据库。

（3）右键单击“数据库”节点，在弹出的快捷菜单中选择“新建数据库”命令，将弹出一个“数据库属性”对话框，如图 6-28 所示。

（4）在“常规”选项卡中，可以在“名称”文本框中指定数据库名字，如 DBNEMP。

（5）在“数据文件”选项卡中，可以设置数据库文件的名字和存放位置。默认文件名是 DBNEMP，默认位置是 C:\Program Files\Microsoft SQL Server\MSSQL\Data\DBNEMP.mdf，即在 SQL Server 安装目录的\Data 子目录中，如图 6-29 所示。

“初始大小”项用来设定数据库文件初始大小，默认值是 1MB。这个值可根据自己的网站规模大小来决定。

（6）在如图 6-30 所示“事务日志”选项卡中，可以进行如下设置。

① 在“事务日志文件”选项区域中，可以设置事务日志文件的文件名、位置和初始大小。

② 在“文件属性”选项区域中，选中“文件自动增长”复选框，可以让 SQL Server 服务



图 6-28　“数据库属性”对话框

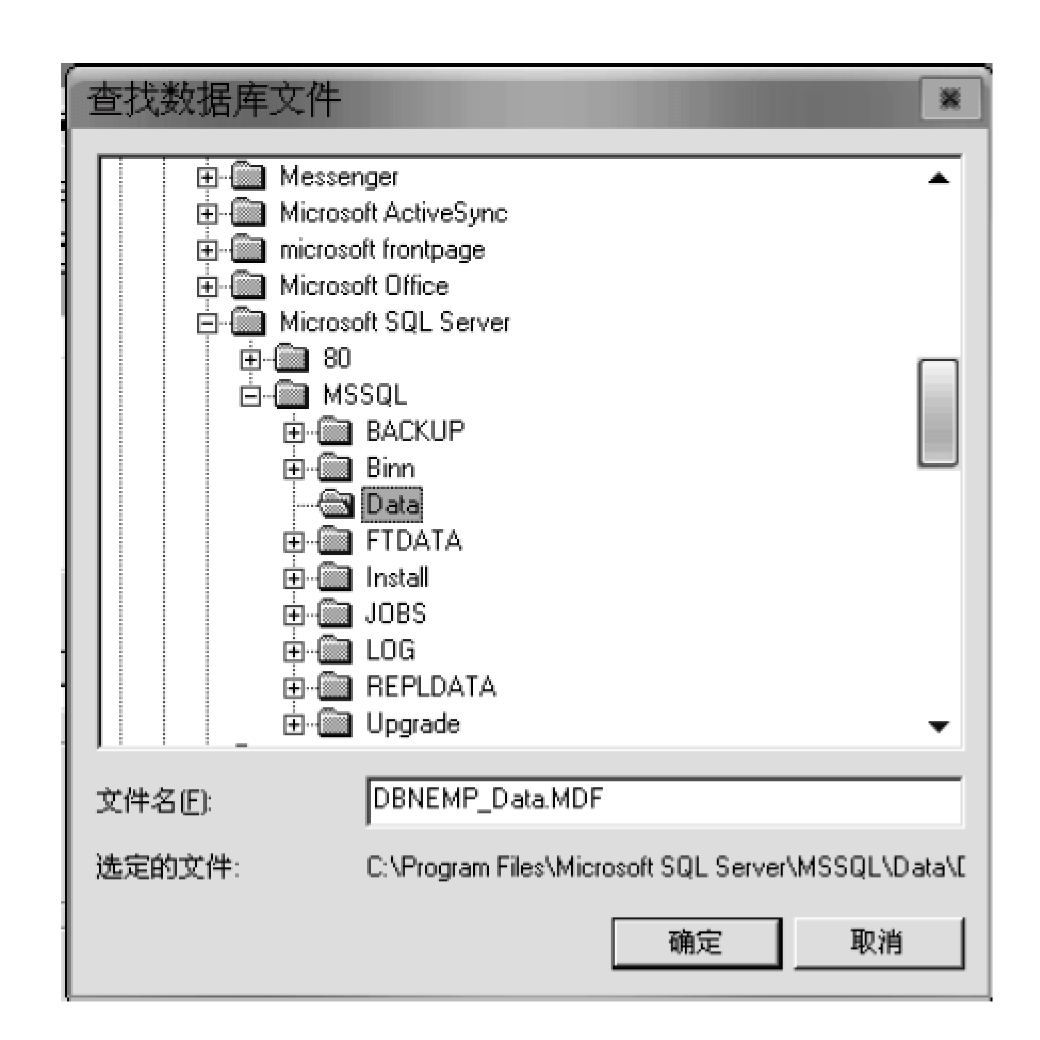


图 6-29　DBNEMP_Data.MDF 的默认位置

器自动增加事务日志文件的大小。

③“文件增长”选项区域用于选择文件自动增长的方式，或者按兆字节增长，或者按百分比增长。

④“最大文件大小”选项区域用于决定事务日志文件的最大尺寸。默认是不限定文件大小。

这样就建立了一个名为 DBNEMP 的数据库，它的各项参数如下。

- 数据库文件为 DBNEMP，保存在 C:\Program Files\Microsoft SQL Server\



图 6-30 “事务日志”选项卡

MSSQL\Data 中。

- 事务日志文件为 DBNEMP_Log.ldf,保存在 C:\Program Files\Microsoft SQL Server\MSSQL\Data 中。
- 文件初始大小是 1MB。
- 每次到达规定尺寸,就自动增加 10%的大小。
- 没有最大尺寸限制。

2. 创建访问 DBNEMP 数据库的新登录

下面将为 DBNEMP 数据库建立一个新登录。建立登录的步骤如下。

在企业管理器左边的树形控件中选择“安全性”节点。

(1) 在右边窗口中出现的“登录”图标上单击右键。

(2) 在弹出的快捷菜单中,选择“新建登录”命令来建立一个新登录,如图 6-31 所示。

选择“新建登录”命令后,将出现如图 6-32 所示的对话框,它有“常规”、“服务器角色”和“数据库访问”三个选项卡。

1) “常规”选项卡

在“常规”选项卡的“名称”文本框中输入新登录的名称,如图 6-32 所示。

“身份验证”选项区域用于指定登录的认证方式。有两种方式,其一为 Windows 身份验证,表示采用 Windows 用户认证。当采用这种方式认证时,只有使用该 Windows 用户登录系统的用户才被 SQL Server 作为合法用户。其二为 SQL Server 身份验证,表示使用 SQL Server 用户认证,SQL Server 为该登录建立一个账号,要求输入密码。

这里采用 SQL Server 身份验证,密码设为 DBNEMP。

在“默认设置”选项区域中,可以设定默认数据库和默认语言。这里设置默认数据库

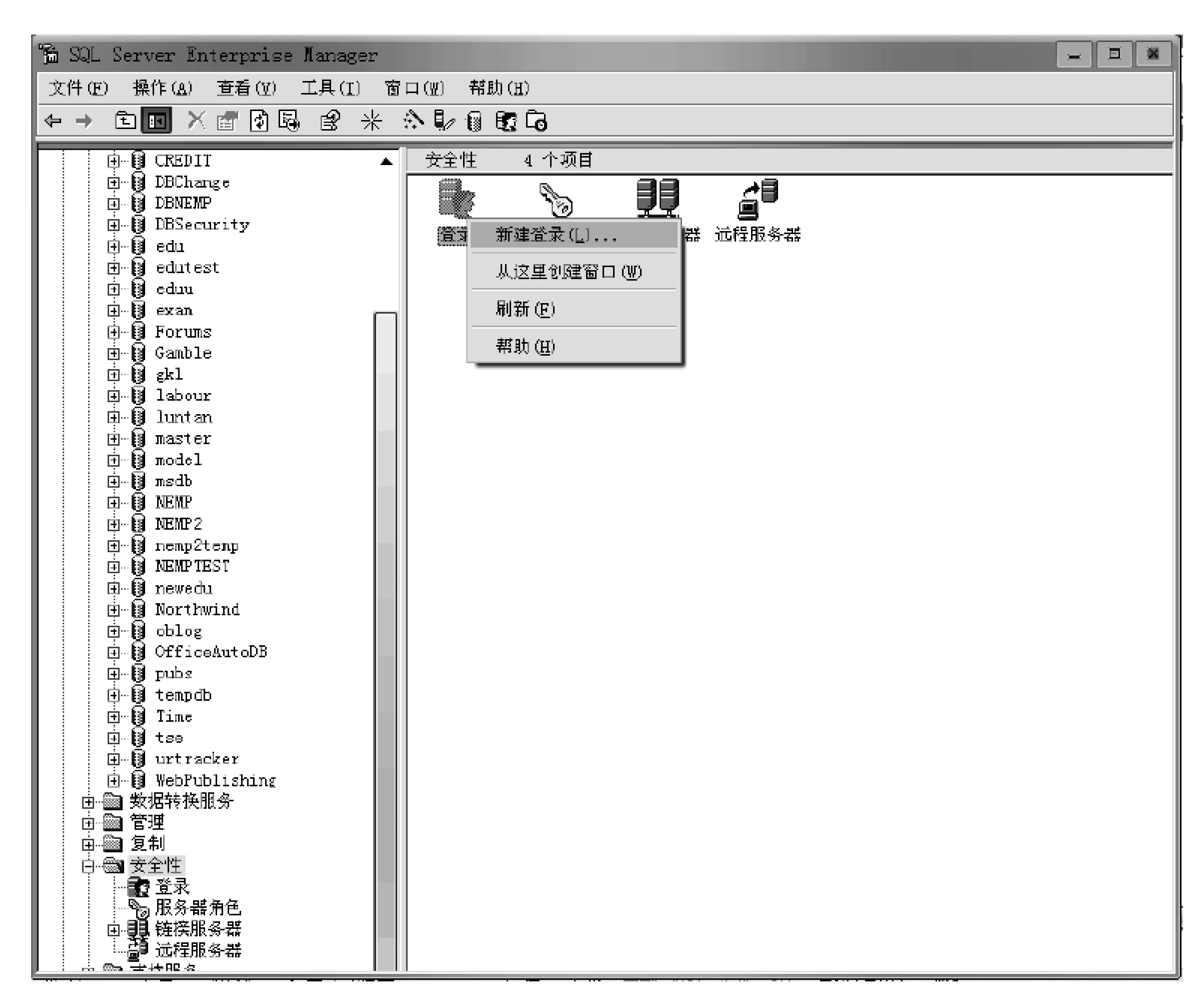


图 6-31　新建登录



图 6-32　“常规”选项卡

为刚才建立的 DBNEMP 数据库，默认语言为“默认”。

2）“服务器角色”选项卡

“服务器角色”选项卡如图 6-33 所示。

在这个选项卡中可以设置登录在服务器范围内的安全特性，它的设置决定了该登录



图 6-33 “服务器角色”选项卡

可以执行哪些操作，如 System Administrators、Security Administrators、Server Administrators 及 Setup Administrators 等。

单击“属性”按钮，出现“服务器角色属性”对话框，在其中可以查看所选定角色的属性，如图 6-34 所示。



图 6-34 “服务器角色属性”对话框

在该对话框的“常规”选项卡中列出了属于该角色的登录。使用者可以在该角色中增加、删除这些登录。在“权限”选项卡中，列出了指定服务器角色所能执行的操作。

图 6-35 所示为 Disk Administrators 可执行的操作。

3）“数据库访问”选项卡

“数据库访问”选项卡如图 6-36 所示。

在“指定此登录可以访问的数据库”列表框中，列出了该 SQL Server 数据库服务器中



图 6-35　“权限”选项卡



图 6-36　“数据库访问”选项卡

所有的数据库，有 DBNEMP、pubs 及 Northwind 等。

将允许此登录访问的数据库名字前的“许可”复选框选中，否则这个登录就无法访问该数据库。这里选择 DBNEMP 数据库。

在“DBNEMP 的数据库角色”列表框中可设定此登录的访问权限，可以是 public、db_owner 或 db_accessadmin 等。

将 public 与 db_owner 选中，使该用户有完全的权限访问控制数据库。

6.2.7　设计 Role 表

在 SQL Server 企业管理器中选中 DBNEMP 数据库下的“表”节点，如图 6-37 所示。这里所列出的表包括 SQL Server 自动生成的表(用于维护数据库，初学者不应随意

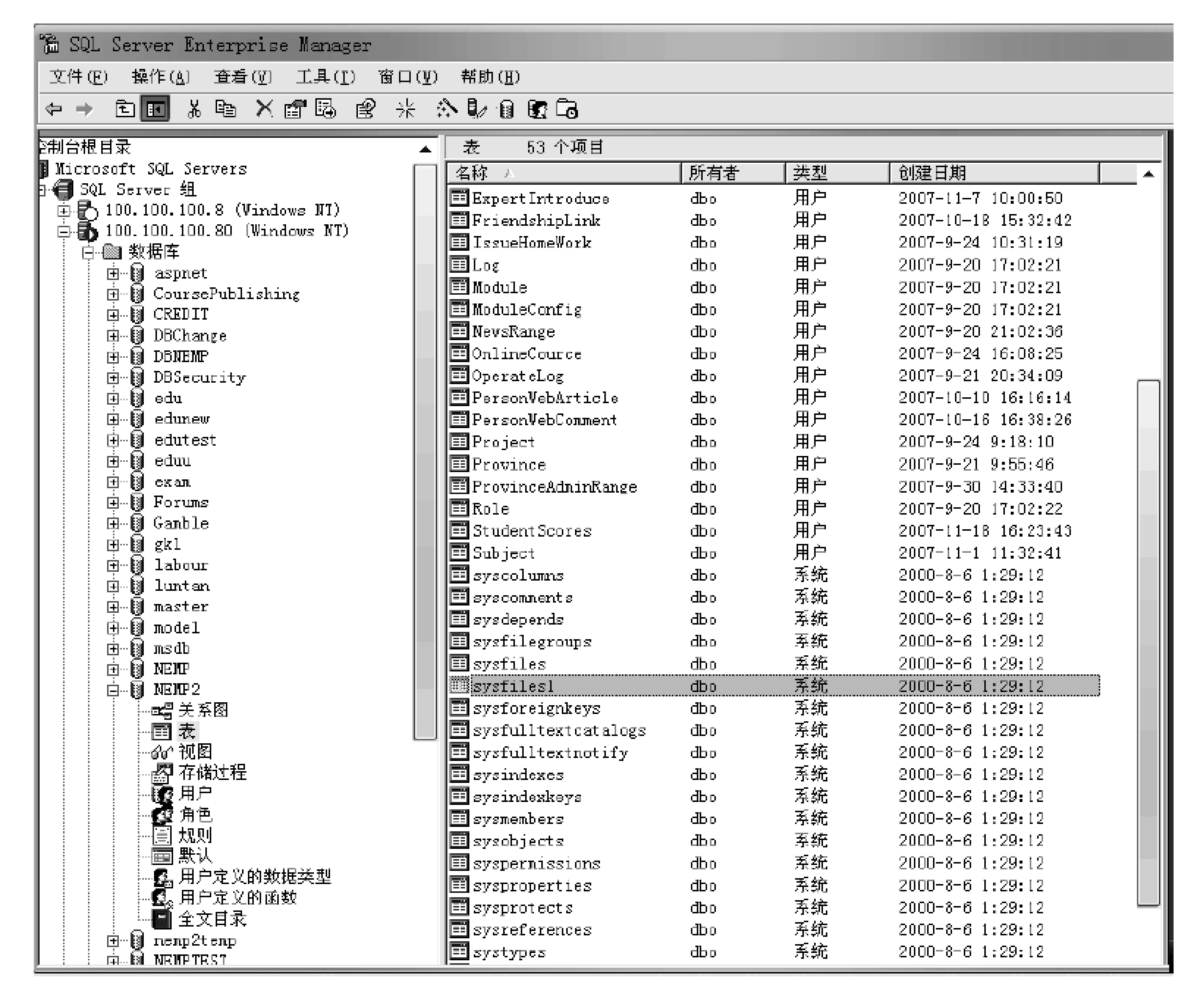


图 6-37 “表”节点

改变)和自定义的表。

现在就开始建立一个表。在右边窗格的任意位置单击右键或右键单击“表”节点,在弹出的快捷菜单中选择“新建表”命令,建立一个新数据表,如图 6-38 所示。

SQL Server 提示输入一个表名,该名字不能与数据库中已有的表名重复。

先建立一个角色表,名字为 Role,如图 6-39 所示。也可以给它一个完全不同的名字,不过一般要符合见名知意原则。

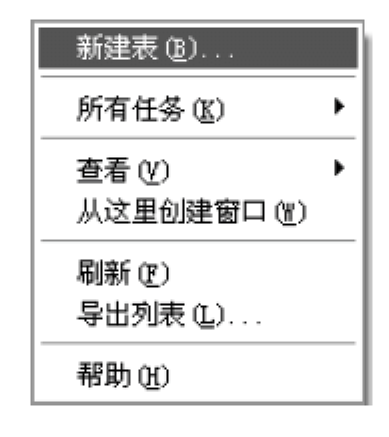



图 6-38 快捷菜单

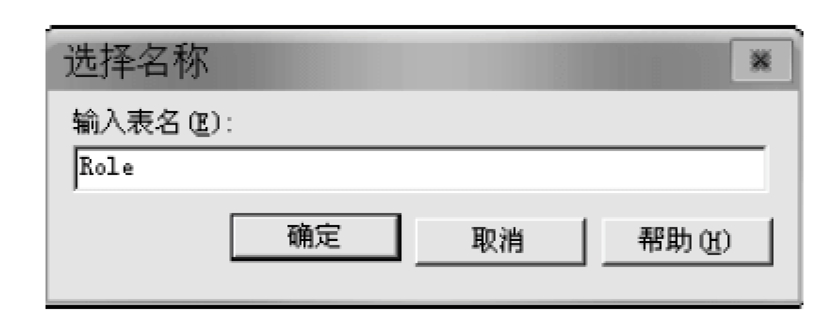


图 6-39 输入表名

输入表名后,SQL Server 就建立了一个名为 Role 的表。

然后对表进行设计,如图 6-40 所示。设定各个列的列名、数据类型、数据长度(所占字节数)、精度(数值型数据的位数)、小数位数、是否允许为空、默认值、标识、标识种子、标识递增量及是否为 RowGrid。

将第一列命名为 RoleId,标识设为“是”,标识种子设为 1,标识递增量也设为 1。这

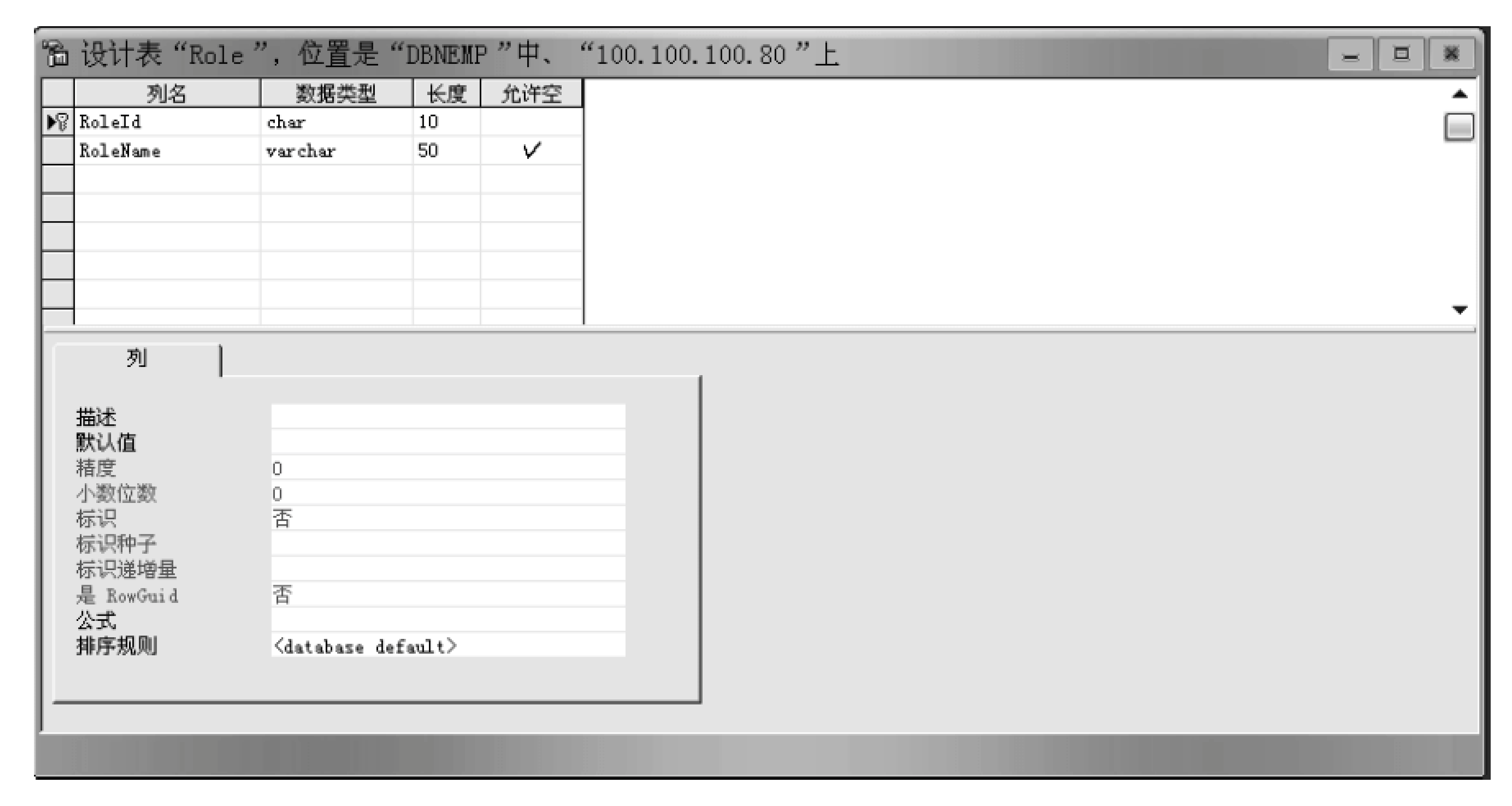


图 6-40　对 Role 表进行设计

样，数据库中每增加一条记录，RoleId 字段就会自动加 1。

RoleId 的数据类型一般可设为 int 型，长度为 4 个字节，也就是可以容纳 4 228 250 625 个用户，这几乎对于任何规模的网站都是足够的。Tiny-int 型数据长为一个字节，只能容纳 256 个用户，这显然不够。

右键单击空白处，在弹出的快捷菜单中选择“属性”命令，弹出表的“属性”对话框，如图 6-41 所示。

在此“属性”对话框中，可以更改表名，如将 Role 改成任何其他名字；可以改变表文件组和文本文件组，默认的值为 PRIMARY；还可以建立、管理对表和列的检查约束。

再打开“索引/键”选项卡，如图 6-42 所示。

图 6-41　表的“属性”对话框

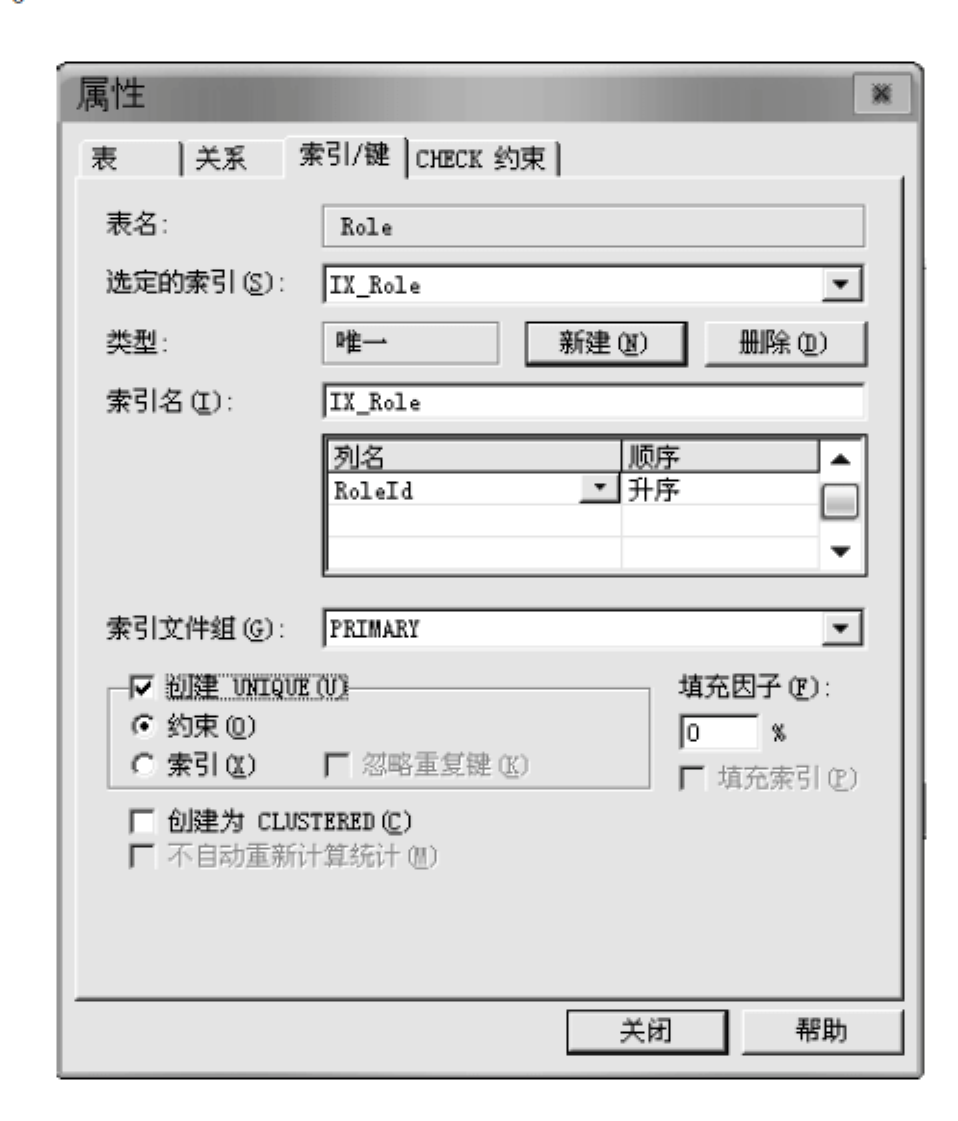


图 6-42　“索引/键”选项卡

在“选定的索引”下拉列表中选择 PK_Role 项，可以看到“类型”栏里的“主键”变成了“索引”。单击“新建”按钮，从“列名”下拉列表中选择 RoleId 作为索引。这样，查询数据库时，速度会因建立了索引而加快。

这样 Role 表就设计好了。Role 表的各个字段含义比较简单，读者可以理解。

按照上述方法，再分别建立其他的表。

6.3 发布站点

6.3.1 发布 Web 页

要想在 Web 站点上发布信息，首先应建立 Web 站点，同时指明哪些目录包含要发布的文档。Web 服务器无法发布未在这些指定目录中的文档。所以，配置 Web 站点的第一步是确定文件的组织方式。可以使用 Internet 信息服务管理器确定哪些目录是站点的一部分。当安装 Windows Server 2000 时，IIS 将创建默认的 Web 站点。如果希望马上开始而不创建特殊的目录结构，并且文件全部位于运行 Internet 信息服务的计算机上，那便可以立刻将 Web 文件复制到默认的主目录 C:\InetPub\wwwroot 来发布文档(对于 FTP 站点，应将文件复制到 C:\InetPub\Ftproot)。内部网用户可以输入“http://服务器名/文件名”访问到这些文件。

1. 在默认 Web 站点上发布内容

(1) 为 Web 站点创建主页。

(2) 将主页文件命名为 Default.aspx。

(3) 将主页复制到 IIS 的默认 Web 发布目录中。默认 Web 发布目录也称为主目录，安装程序提供的位置是\InetPub\wwwroot。

(4) 如果网络具有名称解析系统(通常为 DNS)，那么访问者可以简单地在其浏览器地址栏中输入 Web 站点名来访问站点。如果网络没有名称解析系统，那么访问者必须输入 Web 站点的数字 IP 地址。

2. 定义主目录

每个 Web 站点必须有一个主目录。主目录位于所发布网页的核心位置。它包含带有欢迎内容的主页或索引文件，并且包含到所在站点其他网页的链接。主目录映射为站点的域名或服务器名。例如，如果站点的 Internet 域名是 www.microsoft.com，并且主目录是 C:\Website\Microsoft，那么浏览器将使用 URL http://www.microsoft.com 访问主目录中的文件。在内部网上，如果服务器名是 MADIS-lhf，浏览器将使用 URL http://MADIS-LHF 访问主目录上的文件。

6.3.2 发布站点

如果一切就绪并编译成功，使用 Visual Web Developer 发布 Web 站点。具体步骤如下。

(1) 在 Microsoft Visual Studio2005 中打开本项目 NEMPOffice(D:\ NEMPOffice)，选择

菜单“网站”→“复制网站”(如图 6-43 所示)。



图 6-43　复制网站页面

(2) 单击连接：连接到框，并在文本框中输入 NEMPOffice，如图 6-44 所示。



图 6-44　“查找数据库文件”对话框

要想在远端实际部署站点，要选择左端菜单中的远程站点，然后提供站点位置的 URL 或 IP 地址，可能还要输入相关用户 ID 和密码。不幸的是，对于已经部署站点的读者不能测试站点的 Web 空间。

(3) 单击“打开”按钮，并在弹出窗口提问是否要创建一个新文件夹时单击 Yes 按钮。

(4) 选择所有文件，如图 6-45 所示。

(5) 单击蓝色向右箭头按钮复制文件，如图 6-46 所示。



图 6-45 建立远程连接



图 6-46 复制

(6) 选择“文件”→“关闭”命令,关闭站点。

(7) 选择“文件”→“打开”→“网站”命令,然后选择 NEMPOffice。

(8) 运行新的 Web 站点,如图 6-47 所示。



图 6-47　更新后的 Web

第 7 章　网站安全和维护

在计算机系统中，网站是最容易受到攻击的对象，因为网站本来就是暴露在 Internet 中让用户访问的，所以受到攻击不可避免，关键是如何提高自身的抗攻击能力。在面临网络电子商务的全面开展之际，网站的安全问题将成为建设大型电子商务系统的一个非常关键的问题。本章从网站所面临的安全问题入手，介绍一些提高网站安全性能的措施和手段。

7.1　网站安全

网站的安全包括网站的系统安全和网站的信息安全。如果是租用的虚拟主机或申请的网页空间，那么安全的主要工作由提供商承担，但还是有许多问题需要考虑。如果是拥有自己的服务器，那么就必须全面地考虑网站安全的问题。

网站安全的内容包括网站实体的安全，如计算机机房的物理条件和设施、计算机硬件和网络设施及附属设备等；网站软件的安全，保证网站不受计算机病毒的侵害、不被非法侵入、网站软件不被非法复制篡改等；网站中数据的安全，对于网站中重要的数据不被非法存取，保证数据的完整性等；网站的安全管理，如建立安全的管理制度、对突发事件的应急措施等。

要做好网站的安全应该从影响网站的安全因素入手。首先是网站系统软件的选择，以保证其自身的安全。对网站中比较重要的数据，分级别存放，针对不同的用户开放不同的共享级别。对于较大网站的安全管理，可设置一定的安全管理规范，配置安全管理员等。还有一个影响网站安全的因素就是病毒和黑客的入侵，这可能是对网站危害最大的一个因素，防止此类危害的一个有效手段就是使用防火墙技术。

所谓防火墙其实是位于服务器上的一种程序，它是网络上的一种隔离控制技术，可在某网站和外界不安全网站之间设置障碍。它具有双向的功能，既可阻止外界对信息资源的非法访问，也可以防止站内信息的非法输出。

7.1.1　网站所面临的安全问题

要提高网站的安全性能，必须了解网站面临的安全问题，需要对网络信息系统存在的安全缺陷和可能受到的各种攻击有深入和正确的理解，正所谓知己知彼，才能百战不殆。

当评判一个网站是否面临安全问题时,要从多方面了解各个组件的弱点是什么,包括硬件和软件平台,因为安全性的高低取决于最薄弱环节,符合"水桶理论",即弱的部分决定了整体安全性能,而且黑客攻击也是从薄弱的环节入手。通过近几年在 Internet 上发生的黑客攻击事件可以看出,威胁网络安全的基本模式都很相似,特别是在大量黑客软件工具出现以后,使得网站的安全受到更严重的威胁。

隐藏在世界各地的攻击者通常可以利用网站所有可能的错误,包括设计错误、安装配置错误等,向网络发起攻击。系统软件存在的漏洞是最大隐患,因为攻击者只要找到某一种系统软件的安全漏洞,那么他就可以攻击所有使用这种系统软件的网站,所以及时地为系统软件做安全补丁是很重要的。

除了自然灾害外,通常将网络的安全漏洞分为两大类。

1. 软硬件系统平台的安全漏洞

其包括无意的漏洞和有意的漏洞。无意的漏洞是系统设计过程中不经意的失误或欠深入考虑而造成的,例如主机存放的环境问题、物理防范措施不当等;有意的漏洞是某些系统设计人员专门开的后门,虽然开始不为人知,可是时间长了一旦泄露出去,后果就不堪设想。硬件方面如 PIII 芯片的序列号问题,软件方面如微软的设计人员曾在 ASP 页面中开了查看源文件的后门,这个漏洞需要安装一些 SP 补丁包才能解决。

2. 自身的人为安全隐患

不管是有意还是无意的,人为因素往往对网站安全有重要影响。例如操作员安全配置不当造成的安全漏洞;用户安全意识不强,口令选择不慎,将自己的账号随意转借他人或与别人共享等都会对网络安全带来威胁。

1) 系统平台的安全

首先 Internet 这个大网络平台从建立开始就缺乏总体的安全构想和设计,而且 TCP/IP 协议是在可信的环境下专门对网络互联设计的,缺乏全面安全措施的考虑,所以 Internet 的网络安全就成为互联技术发展的一个关键因素。在这样的大环境下,就要加强自身系统的防范。建立网站必须要有硬件和软件平台,其中可能包括主机、防火墙、路由器、防病毒墙、操作系统、Web 服务器、电子邮件服务器和数据库等,其中任何一个方面的安全缺陷都有可能对网站的安全造成潜在的危险。

对于系统平台的安全,一般需要考虑以下问题。

(1) 确认服务器、操作系统和其他系统软件是否有安全漏洞,如果有,是否有相应的措施预防。对于硬件,确认是否要更换部件,软件是否要安装补丁程序等。

(2) 采用适当的措施保护 Web 服务器主机系统不会受到外网或内网的攻击。

(3) 保护 Web 服务器免于拒绝服务的攻击。

(4) 采用一次性登录方式,减少账户管理的复杂度,简洁的账户系统更容易实施安全策略,对过期不用的账户应及时删除。

(5) 建立一个可持续周转的安全策略,并确保策略的实施。

(6) 注意网站安全日志的记录与查看,因为绝对安全的系统是不存在的,安全的网站和不安全的网站也是相对而言,所以安全的网站抗攻击能力较强,攻击者可能会花费很大

的精力和时间才能攻破。在这段时间里，如果能随时查阅安全日志，发现有攻击存在时，采取必要的措施就可以使攻击者退却。

一个安全的系统至少要做到当某一部分被攻破时，其影响范围应该尽量最小化，避免安全问题进一步扩散，例如即使泄露一个文件的密钥，攻击者也不能解开硬盘上所有的加密文件。作为一个优秀的系统设计者，必须考虑入侵发生后的应对办法，并且在设计时还应该设想一下如果系统被攻破应该怎么办。

攻击者攻破了某个网络目录服务器，由此将对某些服务器的访问定向到攻击者指定的机器，该机器可能中断客户和服务器之间来往的数据，使客户和服务器双方都没有意识到第三方的存在，很多敏感数据就此被窃取。这种方式通常被称为“中间人”攻击。

在内部网中，当各工作站使用共享文件系统共享可执行文件时，一个内部攻击者可能监听了工作站向文件服务器发出的文件访问请求，抢先把修改过的文件块传给工作站。由于内部网缺少判断数据包发送地址的能力，工作站会接收该数据包，将真正的包当做重复数据丢弃，从而工作站将毫无察觉地使用修改后的程序。这类错误应该算一个比较严重的问题，当前许多操作系统在这方面没有做很大的改进。Windows NT 的 Service Pack 3 虽然注意到了这个问题，但在同其他系统的互操作性方面却受到了影响。

2）自身的人为因素

许多系统被攻破是因为它们严重依赖于用户创建的口令，为了便于记忆，用户通常选择简单容易的字符串而不选择复杂性很强的口令，即使该口令被加密，比起随机生成的密码也更容易破解。此外，为了工作上的方便，有些用户可能会把自己的口令告知同事，从而有可能散布开。通常负责密钥管理和系统维护的人越多，系统出问题的可能性就越大。对此，最好的解决办法是建立完整的安全策略，严格控制用户的权限及对关键信息的访问。对于银行之类的系统，必要时可对某些关键数据采用秘密共享机制的保护策略，以保证在若干个人同时在场的情况下才能启用。与此同时，还要加强审计功能，保证每一件事务都有据可查。安全系统应该能够提供解决这一问题的机制，不允许任何人在系统中留有后门。对于安全因素需要考虑以下问题：确保被保护的 Web 内容只有少数人员被授予特定的访问权限；更换系统维护人员时需要更改关键的口令。

3）攻击方式

对网站的攻击一般都是人为的恶意攻击，这是计算机网络所面临的最大威胁，敌手的攻击和计算机犯罪就属于这一类。此类攻击又可以分为以下两种：一种是主动攻击，它以各种方式有选择地破坏信息的有效性和完整性；另一类是被动攻击，它是在不影响网络正常工作的情况下，进行截获、窃取和破译重要机密信息。这两种攻击均可对计算机网络造成极大的危害，并导致机密数据的泄露。

常见的攻击方法分为以下几种类型：试探、扫描、窃取用户账号、获得超级用户权限、数据包窃听、拒绝服务、利用信任关系、恶意代码（如特洛伊木马、病毒和“蠕虫”等）及攻击 Internet 基础设施（如 DNS 系统和网络路由等）。

一般来说，攻击者对目标进行攻击要经历三个步骤：情报搜集、系统的安全漏洞检查和实施攻击。情报搜集的目的是为了得到所要攻击的目标系统的相关信息，为进一步行动做好准备。攻击者可以利用公开的协议或工具，也可以利用一切可以获得的情报，收集

目标网络系统中的各个主机系统的相关信息。这些公开协议及工具包括 HTTP、FTP、SMTP 及 Telnet 等，网络服务可能会暴露操作系统类型和版本等信息。SNMP 协议、Traceroute 等工具可以用来检查网络系统路由表、到达目标主机所要经过的网络和有关路由信息细节。WHOIS 协议、DNS 服务器及 NetBIOS 协议等可能会给出目标主机名称等信息。不同 TCP/IP 协议的实现会产生不同的“指纹”，可用于判断目标主机所使用的操作系统。一般来说，在使用自动化工具时，这两步可以同时进行。当攻击者使用上述方法，收集或探测到一些有用信息之后，就可以对目标系统实施攻击。

攻击者一旦获得了对攻击目标系统的访问权后，又可能有多种选择。可以进一步发现被攻击系统在网络中的信任关系，这样攻击者就可以通过信任关系展开对整个系统的攻击；也可以在目标系统中安装探测器软件，包括特洛伊木马等，用来窥探所在系统的活动，搜集攻击者感兴趣的信息，如 Telnet、FTP 及 POP3 的账号和口令等。如果攻击者在被攻破系统上获得了特权用户权限，就可以读取邮件、搜索和盗窃私人文件、毁坏重要数据及破坏整个系统的信息等，造成极其严重的后果。此外，还可以清除日志，毁掉入侵痕迹。在被攻破的系统上建立后门，以便在先前的攻击被发现后，还可以继续访问这个系统。

网络攻击与反攻击的双方其实是极其不平等的。对攻击者来说，只要发现一点漏洞就可以实施不法手段；而对反攻击者来说，需要考虑到系统的方方面面。系统的安全应该是“系统的”安全，是全方位的安全，需要安全产品生产者和用户双方做出很大的努力才能达到。

7.1.2　网站安全的含义

网站安全是指利用网站管理的控制和技术措施，保证在一个网站环境里，信息数据的机密性、完整性及可使用性受到保护。

网站安全的主要目标是要确保经网站传送的信息，在到达目的站时没有任何增加、改变、丢失或被非法读取。要做到这一点，就必须保证网站系统软件、应用软件系统和数据库系统具有一定的安全保护功能，并保证网站部件如终端、路由器和防火墙等的功能不变，而且仅仅是那些被授权的人可以访问。

7.1.3　网站安全的内容

网站的安全性问题实际上包括两方面的内容，一是网站的系统安全，二是网站的信息安全，而保护网站的信息安全是最终目的。就网站信息安全而言，首先是信息的保密性，其次是信息的完整性。另一个与网站安全紧密相关的概念是拒绝服务。

网站拒绝服务的表现是什么？

拒绝服务是与网站系统可靠性有关的一个重要问题，主要包括 3 个方面的内容：系统临时降低性能，系统崩溃而需人工重新启动，因数据永久性丢失而导致较大范围的系统崩溃。

网站安全的内容大致包括以下 4 个方面。

1. 网站实体安全

如计算机机房的物理条件、物理环境及设施的安全标准，计算机硬件、附属设备及网站传输线路的安装和配置等。

2. 软件安全

如保护网站系统不被非法侵入，系统软件与应用软件不被非法复制、篡改及不受病毒的侵害等。

3. 网站中的数据安全

如保护网站信息的数据不被非法存取，保护其完整和一致等。

4. 网站安全管理

如对运行时突发事件的安全处理等，包括采取网络安全技术、建立安全管理制度、开展安全审计及进行风险分析等。

7.1.4 网站的安全性能

网站的安全性能主要包括保密性、完整性和可使用性三个方面。

1. 保密性

指网站中有保密要求的信息只能够经过允许的人员以经过允许的方式使用。从技术上说，任何传输线路，包括电缆、光缆、微波和卫星等都有可能被窃听。

可以窃听双绞线或同轴电缆中传输的信号吗？微波也能被窃听吗？

对于双绞线或电缆的窃听，可以是接触式的；也可以是非接触式的，即通过电磁感应或利用电磁辐射来窃听。另外，现在已有灵敏度很高的TEMPEST设备，即使像微波那样的视距传播，由于其波束有一定的宽度，因此天线也可以放在射线中央以外的地区接收。

2. 完整性

指网站中的信息安全、精确和有效，不因种种不安全因素而改变信息原有的内容、形式和流向。造成信息完整性被破坏的原因可分为人为和非人为两种。非人为因素如通信传输中的噪声干扰，系统硬件或软件的差错等。人为因素包括有意和无意两种，前者如非法分子对计算机的侵入、合法用户越权对网站内数据的处理，以及隐藏的破坏性程序等；后者如操作失误或使用不当等。对于大多数网站来说，计算机病毒、时间炸弹和逻辑陷阱等都属于隐藏的破坏性程序，能破坏数据信息。对信息完整性的破坏是对网站安全的主要危害。

3. 可使用性

指网站资源在需要时即可使用，不因系统故障或误操作等使资源丢失或妨碍对资源的使用。网站可使用性还包括具有在某些不正常条件下继续运行的能力。对网站可使用性的影响包括合法的用户不能正常访问网站的资源，和有严格时间要求的服务不能得到及时响应等。影响网站可使用性的因素包括人为与非人为两种。前者如非法占用网站资源、切断或阻塞网站通信，以及病毒或“蠕虫”降低网站性能，甚至使网站瘫痪等。后者如

灾害事故和系统死锁、系统故障等。

7.1.5　网站安全因素

网站的安全因素是多方面的。从网站组成结构上分为计算机信息系统的，有通信设备和设施的；从内容上分有技术上的，有管理上的；从管理上分有内部的和外部的等。具体来说主要有以下几个方面的问题。

1. 网站系统软件自身的安全问题

网站系统软件的自身安全与否直接关系到网站的安全。网站系统软件的安全功能较少或不全，以及系统设计时的疏忽或考虑不周而留下的“破绽”，都等于给危害网站安全的人和事留下了许多“后门”。

2. 网站系统中数据的安全问题

网站中的信息数据是存放在数据库中的，通常也指存放在服务器中的信息集合，供不同的用户来共享。数据库存在着不安全性和危险性，因为在数据库系统中存放着大量重要的信息资源，在用户共享资源时可能会出现以下现象，即授权用户超出了他们的访问权限进行更改活动，非法用户绕过安全内核窃取信息资源等。因此提出了数据库安全问题，也就是要保证数据的安全可靠和正确有效。对数据的保护主要是指针对数据的安全性、完整性和并发控制三方面进行保护。

数据的完整性是指防止数据库存在不符合语义的数据，以及防止由于错误信息的输入、输出而造成无效操作和错误结果。并发控制即数据库是一个共享资源，在多个用户程序并行地存取数据库时，就可能会产生多个用户程序通过网站并发地存取同一数据的情况，若不进行并发控制就会使取出和存入的数据不正确，破坏数据库的一致性。

3. 传输线路的安全与质量问题

尽管在同轴电缆、微波或卫星通信中要窃听其中指定一路的信息是很困难的，但是从安全的角度来说，没有绝对安全的通信线路。

4. 网站安全管理问题

网站安全管理最重要的是从体系结构上要实行最小权限原则。世界上现有的信息系统绝大多数缺少安全管理员，缺少信息系统安全管理的技术规范，缺少定期的安全测试与检查，更缺少安全监控。我国许多企业的信息系统已经使用了许多年，但计算机的系统管理员与用户的注册大多还是处于默认状态。

另一方面，也可以说网站的安全问题是天生的，这是由于“整体大于部分之和”的原因。网站由各种服务器、工作站和终端等集群组成，所以整个网站天然地继承了它们各自的安全隐患。各种服务器各自运行着不同的操作系统，各自继承着自身系统的不同安全特性。随着计算机及通信设备组件数目的日益增大，积累起来的安全问题会越来越多。

7.1.6　攻击过程与类型分析

典型的攻击步骤如下。

1. 窥视

相当于通常的窃贼在远处张望一家民居一样，黑客首先利用一些网络工具来确定这些系统在 Internet 上的位置和结构，发现目标系统的外围安全设备类型和结构，并确定侵入点。

2. 外围侦察

相当于现实当中的窃贼打开门锁。已经确定目标和侵入点的网络入侵者，通过外围安全设备的薄弱环节进入网络的正常服务，如通过电子邮件系统和主页系统进入网络。

3. 寻找内部落脚点

一旦侵入者获得了进入网络的权利，那么下一步便是在外围设备中为自己寻找一个安全的、不易被发现的落脚点。通常，侵入者会找到一个能够获得用户权限的主机作为落脚点。落脚点确定后，外部入侵者就变成了系统内部人员，这时入侵者会在系统内部寻找可盗窃的财产和可破坏的目标系统。

主要的破坏动作包括偷窃软件源代码和财政金融数据，访问机密文件，破坏数据或硬件，还包括安置为日后再次侵入做准备的“特洛伊木马”。

破坏动作完成后，侵入者必须掩盖自己的踪迹，以防被发现。典型的做法是删除或替换系统的日志文件。

侵入者对目标系统的伤害主要有以下几个方面。

1. 非法使用资源

包括对计算机资源、网络连接服务或电话服务等资源的滥用和盗用。通过对系统的控制，侵入者能够无限制地使用这些资源而不必付出任何费用。最典型的是免费使用电话系统或在全球的数据通信网络中无节制地漫游。这种攻击很少造成结构性的损害，但其高昂的费用会转嫁到用户或服务商的头上。

2. 恶意破坏

这种行为会对系统或数据文件造成破坏，也可能会造成停止对合法用户提供服务。典型的恶意破坏包括毁坏数据、修改页面内容或链接。这种破坏有时无需侵入网络，传送进入网络中的文件可附带有破坏性的病毒，对网络设备的信息轰炸也可造成服务中断。网络服务商最容易受到此类攻击。

3. 盗窃数据

随着电子商务的广泛开展，盗窃行为的可能性和危险性也在不断增加，任何有价值的内容都有可能被盗，从数据、服务到整个数据库系统、金融数据和敏感的个人信息等。据 FBI 的估计，每年由于此类数据被盗而造成的损失高达 75 亿美元。

4. 敲诈勒索

由于对破坏系统、盗窃数据或破坏网络完整性的恐惧，企业因此每年都要受到上百美元的勒索。典型的勒索方法是在目标网络中安置特洛伊木马程序，如被勒索者不付款，破坏程序便会被启动。

既然已经知道了攻击者的行为特点和攻击类型，那么一个有效的网络安全方案就必须针对网络的易受攻击点而制定保护措施，同时还要加强对网络的维护。一个系统仅靠外围安全设备的保护是不够的。典型的安全设备，如传统的防火墙、认证设备及系统外壳

无法有效阻挡侵入者，它们只能保护网络的入口，一旦被侵入者攻破便无法检测到来自系统内部的破坏和攻击行为。

管理者应尽量缩短对破坏的反应时间，一旦给攻击者充足的时间，那么任何系统都有可能被攻破。为了减少由于入侵而造成的损失，安全系统应该进行实时的入侵检测，对攻击行为进行扫描和拦截。这种方案可以在被侵入的最初几秒钟之内探测到危险动作。安全系统应对侵入做出快速反应，能够自动阻挡入侵者的破坏行为。解决的办法是为网站安装入侵检测系统。

另外，防止内部人员的攻击也很重要。据估计，有 80% 的攻击来自于内部人员。任何安装系统都需要安装、维护和更新。为了降低开支，理想的安全系统应具备灵活性、可扩充性和透明性。

7.1.7　网站安全策略

1. 物理安全策略

目的是保护计算机系统、网络服务器及打印机等硬件和通信链路免受自然灾害、人为破坏和搭线攻击等。采用的措施有验证用户的身份和使用权限、防止用户越权操作；确保计算机系统有一个良好的电磁兼容工作环境；建立完备的安全管理制度，防止非法进入计算机控制室和各种偷窃、破坏活动的发生。

2. 访问控制策略

访问控制是网络安全防范和保护的主要策略，主要任务是保证网络资源不被非法使用和访问。它也是维护网络系统安全、保护网络资源的主要手段。各种安全策略必须相互配合才能真正起到保护作用，但访问控制可以说是保证网络安全最重要的核心策略之一。

访问控制策略主要包括以下几个方面。

1）入网访问控制

入网访问控制为网络访问提供第一层访问控制，控制哪些用户能够登录到服务器并获取网络资源，控制准许用户入网的时间和准许他们在哪台工作站入网。

用户的入网访问控制可分为三个步骤：用户名的识别与验证、用户口令的识别与验证、用户账号的默认限制检查。三道关卡中只要任何一关未过，该用户便不能进入该网络。对网络用户的用户名和口令进行验证是防止非法访问的第一道防线。用户登录时首先输入用户名和口令，服务器将验证所输入的用户名是否合法。如果验证合法，才继续验证用户输入的口令，否则用户将被拒之于网络之外。用户的口令是用户入网的关键所在。为保证口令的安全性，用户口令不能显示在显示屏上，口令长度应不少于 6 个字符，字符最好是数字、字母和其他字符的混合。用户口令必须经过加密，加密的方法很多，其中最常见的方法有基于单向函数、测试模式、公钥加密方案、平方剩余、多项式共享和数字签名方案的口令加密等。经过上述方法加密的口令，即使是系统管理员也难以得到它。

用户还可采用一次性用户口令，也可用便携式验证器（如智能卡、磁卡和安全密码发生器等）来验证用户的身份。网络管理员应该可以控制和限制普通用户的账号使用、访问网络的时间和方式。用户名或用户账号是所有计算机系统中最基本的安全形式。用户账号应只有系统管理员才能建立。用户口令应是每位用户访问网络所必须提交的“证件”。

用户可以修改自己的口令,但系统管理员应该可以控制口令的最小长度,强制修改口令的时间间隔、口令的唯一性和口令过期失效后允许入网的宽限次数等。用户名和口令验证有效之后,再进一步履行用户账号的默认限制检查。

2）网络的权限控制

网络的权限控制是针对网络非法操作所提出的一种安全保护措施。用户和用户组被赋予一定的权限,网络控制用户和用户组可以访问哪些目录、子目录、文件或其他资源,指定用户对这些文件、目录和设备能够执行哪些操作。受托者指派和继承权限屏蔽可作为其两种实现方式。受托者指派控制用户和用户组如何使用网络服务器的目录、文件和设备。继承权限屏蔽相当于一个过滤器,可以限制子目录从父目录那里继承哪些权限。可以根据访问权限,将用户分为特殊用户(即系统管理员)、一般用户和审计用户(负责网络的安全控制与资源使用情况的审计)几类。

3）目录级安全控制

网络应允许控制用户对目录、文件和设备的访问。用户在目录一级指定的权限不仅对所有的文件和子目录有效,还可进一步指定对目录下的子目录和文件的权限。对目录和文件的访问权限一般有 8 种:系统管理员权限、读权限、写权限、创建权限、删除权限、修改权限、文件查找权限和存取控制权限。用户对文件或目标的有效权限取决于用户的受托者指派、用户所在组的受托者指派及继承权限屏蔽取消的用户权限三个方面。一个网络系统管理员应当为用户指定适当的访问权限,这些访问权限控制着用户对服务器的访问。这 8 种访问权限的有效组合可以让用户有效地完成工作,同时又能有效地控制用户对服务器资源的访问,从而加强了网络和服务器的安全性。

4）属性安全控制

当使用文件、目录和网络设备时,网络系统管理员应给文件和目录等指定访问属性。属性安全控制可以将给定的属性与网络服务器的文件、目录和网络设备联系起来。属性安全在权限安全的基础上提供更进一步的安全性。网络上的资源都应预先标出一组安全属性。用户对网络资源的访问权限对应一张访问控制表,用以表明用户对网络资源的访问能力。属性设置可以覆盖已经指定的任何受托者指派和有效权限。属性往往能控制以下几个方面的权限:向某个文件写数据、复制一个文件、删除目录或文件、查看目录和文件、执行文件、隐含文件和安全规则设定共享等。网络的属性可以保护重要的目录和文件,防止用户对目录和文件的误删除、修改和显示等。

5）网络服务器安全控制

网络服务器的安全控制包括可以设置口令锁定服务器控制台,以防止非法用户修改、删除重要信息或破坏数据,以及可以设定服务器登录时间限制、非法访问者检测和关闭的时间间隔等。网络服务器安全控制就是在服务器控制台上执行一系列操作。

6）网络监测和锁定控制

网络管理员应对网络实施监控,服务器应记录用户对网络资源的访问情况。对非法的访问请求,服务器应以图形、文字或声音等形式报警,以引起网络管理员注意。如果不法之徒试图进入网络,网络服务器应能自动记录企图尝试进入网络的次数,如果非法访问的次数达到了设定的数值,那么该账号应被自动锁定。

7）网络端口和节点的安全控制

网络中服务器的端口往往使用自动回呼设备、静默调制解调器加以保护，并以加密的形式来识别节点的身份。自动回呼设备用于防止假冒合法用户，静默调制解调器用以防范黑客的自动拨号程序对计算机进行攻击。网络还常对服务器端和用户端采取控制，用户必须携带证实身份的验证器(如智能卡、磁卡和安全密码发生器等)。在对用户身份进行验证之后，才允许用户进入用户端。然后，用户端和服务器端再进行相互验证。

8) 防火墙控制

防火墙是发展较快、使用较广的一种提供网站安全的技术性措施。它以路由器或堡垒主机的形式建立阻隔内外网的屏障，通过软件设置 IP 通信的安全策略，使网站能在提供服务的前提下还能抵挡外部的侵入。事实上，超过 1/3 的网站是由某种形式的防火墙加以保护的，这是对黑客防范最严、安全性最强的一种方式，任何关键性的服务器都放在防火墙之后。

3. 数据加密策略

加密的目的是保护网内的数据、文件、口令和控制信息，保护网上传输的数据。加密常用的方法有链路加密、端点加密和节点加密三种。链路加密的目的是保护网络节点之间的链路信息安全，端点加密的目的是对源端用户到目的端用户的数据提供保护，节点加密的目的是对源节点到目的节点之间的传输链路提供保护。用户可根据网络情况酌情选择上述加密方式。信息加密过程由形形色色的加密算法来具体实施，它以很小的代价提供很大的安全保护。在多数情况下，信息加密是保证信息机密性的唯一方法。

据不完全统计，到目前为止，已经公开发表的各种加密算法多达数百种。如果按照收发双方密钥是否相同来分类，可以将这些加密算法分为常规密码算法和公钥密码算法两种。在常规密码中，收信方和发信方使用相同的密钥，即加密密钥和解密密钥是相同或等价的；公钥密码的收信方和发信方使用不同的密钥。

常规密码和公钥密码相比较，哪个更安全些？

不能简单地说哪个更好，哪个更安全，它们各有自己的特点。常规密码的优点是有很强的保密强度，且能经受住时间的检验和攻击，但其密钥必须通过安全的途径传送。因此，其密钥管理成为系统安全的重要因素。公钥密码的优点是可以适应网络的开放性要求，且密钥管理问题也较为简单，尤其可方便地实现数字签名和验证。但其算法复杂，加密数据的速率较低。尽管如此，随着现代电子技术和密码技术的发展，公钥密码算法将是一种很有前途的网络安全加密体制。

当然，在实际应用中，人们通常将常规密码和公钥密码结合在一起使用。例如利用 DES 或者 IDEA 来加密信息，而采用 RSA 来传递会话密钥。如果按照每次加密所处理的位来分类，则可以将加密算法分为序列密码和分组密码两种。前者每次只加密一位；而后者则是先将信息序列分组，每次加密一个分组。可以说，如何加密是当前研究的热门话题。

4. 网络安全管理策略

在网络安全中，除了采用上述技术措施之外，加强网络的安全管理，制定有关规章制度，对于确保网络安全和可靠地运行将起到十分重要的作用。网络的安全管理策略包括确定安全管理等级和安全管理范围，制定有关网络操作使用规程和人员出入机房管理制

度，以及制定网络系统的维护制度和应急措施等。

7.1.8 ASP.NET 中的安全性

在 ASP.NET 中讨论安全性首先要解决两个问题，那就是谁有权利进入系统？他进入系统之后能进行何种操作？在解决谁能进入系统的问题中，通常会维护一张允许进入系统的用户的名单，当用户要求进入系统时，判断他是否是合法用户。这样一来，问题就转化为如何有效地判别一个用户是否是系统的有效用户，我们称之为“验证”过程。一个常见的验证过程是，进入某些系统时被要求输入用户名和口令。当用户进入以后，我们只允许他访问事先指定给他的资源，这一过程称为“授权”。只有通过授权检查后，用户才能够对相应资源进行操作。在 ASP.NET 环境中，ASP.NET 和 IIS 结合在一起为用户提供验证和授权服务。

ASP.NET 的应用程序还可以根据进入用户的不同标识，执行相应不同的应用代码。这种方式被称为“角色扮演(impersonation)”。

在一些应用中，对于不同的用户所能够看到和执行的功能是不尽相同的，这就要求 ASP.NET 提供“角色(role)”和“用户(user)”的区分方法。

1. 验证和授权

ASP.NET 和 IIS 一起为用户提供验证服务，用户验证方式有三种，即基本验证方式(basic)、简要验证方式(digest)和窗口验证方式(window)。同时，ASP.NET 支持微软公司的“护照(passport)”验证服务，它单方面提供签到服务和用户描述服务。

此外，ASP.NET 还提供了 Cookies，帮助建立一种基于用户 Form 的验证方式。通过 Cookies，用户的应用程序可以用自己的代码和逻辑实现用户定义的可信性验证。

由于 ASP.NET 的验证服务是建立在 IIS 的验证服务之上的，因此在设立自己的应用服务时有时需要在 IIS 中进行相应的设置。例如，为实现 ASP.NET 的基本验证服务，就必须利用 Internet Service Tool 工具把该应用设置成基本鉴别服务方式。

具体方法如下。

选择“开始”→“程序”→“管理工具”→“Internet 信息服务器”选项，打开“默认 Web 站点”，选择所需的应用站点。

右击鼠标，从弹出的快捷菜单中选择“属性”命令，再从对话框中选择“目录安全性”选项卡。在“匿名访问和验证控制”框中，单击“编辑”按钮。然后在出现的“验证方法”窗口中，选择“验证访问”框中的“基本验证”复选框。最后单击“确定”按钮，就完成了设置基本验证方法。操作的画面如图 7-1 所示。

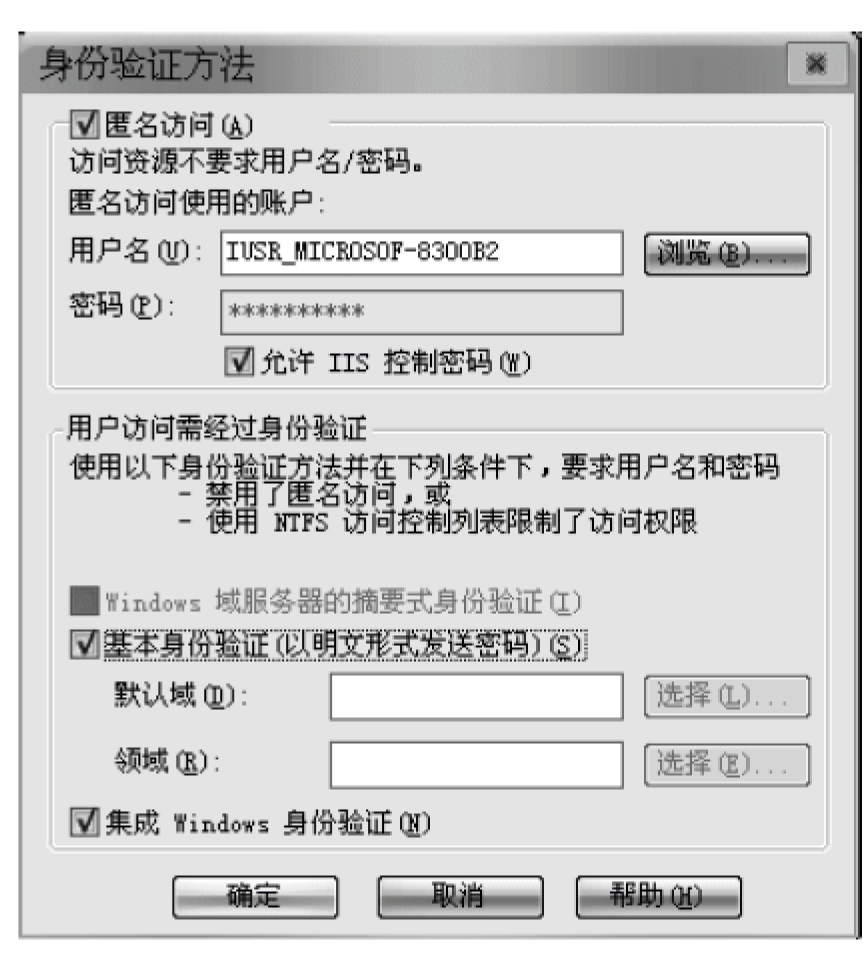


图 7-1 设置验证方式

要使 ASP. NET 的验证机制生效，需要对应用的配置文件 config. Web 进行设置。设置验证需要包含在<security>和</ security >标识之间。它使用<authentication mode="…">语法形式，其中 mode 属性指定了验证采用的方式。具体的验证模式如表 7-1 所示。

表 7-1　ASP. NET 的验证模式

模　式	描　　述
None	没有任何 ASP. NET 的验证服务被激活
Windows	ASP. NET 和 Windows 的机制一起配合使用，可以授权限制 Windows NT 的用户和工作组的访问
Cookie	ASP. NET 验证服务可以管理 Cookies，可以引导使一个未经授权的用户去登录网页。这种模式经常和 IIS 一起配合使用，可以让一个匿名用户去访问应用程序
Passport	ASP. NET 验证服务通过使用护照（passport）服务软件包，附属在服务的外层，更加强了验证服务，这种软件包必须安装到系统中才能使用

举例来说，使用 Cookie 验证方式对某个应用进行验证，那么在其应用的配置文件 config. Web 中，应该包括以下几行。

```
<!--config.Web-->
<configuration>
<Security>
<authentication mode="Cookie"/>
</Security >
</configuration >
```

ASP. NET 提供了两种授权方式：基于 ACL、资源权限的授权方式和 URL 授权方式。基于 ACL 和资源权限的授权方式有点类似于 UNIX 下的文件权限检查，不过它更加严格和完备。当用户请求某个页面时，ASP. NET 检查该页面的 ACL(访问控制列表)和该文件的权限，看该用户是否有权限读取该页面。若有，则该页面称作“已授权”。这种授权方式主要通过系统管理员对文件的权限的设定来实现。

而 URL 授权方式，对于某个用户的页面请求，并不是从文件权限出发，而是根据系统的配置情况，来决定用户的请求是否是经过授权的。URL 授权方式的实现通常是通过设置应用配置文件 config. Web 中关于授权和角色的配置来实现的。

1) 基于 Windows 的验证

当采用基于 Windows 方式的验证后(<authentication mode="windows"/>)，ASP. NET 对每一个页面请求产生一个对象，这个对象可以被 URL 授权方式用来进行授权，或者被用户程序用来判断是否是某种角色。

例如，判断当前用户是否是管理员，可以采用以下的代码。

```
If  User.IsInRole("Administrator")
//如果是管理员组成员,…
    …
Else
    //如果不是管理员组成员,…
…
End If
```

可能用户希望对验证过程加入一些自己的控制代码，那么就需要对 WindowsPrincipal 对象的 WindowsAuthentication_OnAuthenticate 事件做出自己的处理，编写自己的消息处理函数。通常用户通过实现 System. Principal. Iprincipal 类完成。

2）基于 FORM 的验证

基于 FORM 的验证，实际上是允许应用程序定义自己的验证画面和可信性验证。当用户进入时，出现事先定义的画面并要求输入验证要素。输入完毕，用户逻辑对输入进行验证，若通过，则进入应用，否则返回起始输入画面。基于 FORM 的验证，通常都采用 Cookies 技术来实现验证任务。启用基于 FORM 的验证，是在 config. Web 中设置<authentication mode="cookie"/>来实现的。

例如，采用基于 FORM 的验证方式，拒绝匿名用户进入的配置段如下。

```
<!--config.Web-->
<configuration>
        <security>
          <authentication mode="Cookie"/>
            <authorization>
             <deny users="?"/>
           </ authorization>
         </ security>
</ configuration>
```

其中，< authorization >一节中，用标识表示禁止某种用户，“?”代表匿名用户，“*”代表所有用户。当然，在 users 后也可以跟指定的用户，表示只拒绝指定的用户。

此外，在配置节中还有一个 Cookie 标识，它规定了 Cookie 的行为，也较为重要。Cookie 标识有如下三个属性。

- decryptionkey：用于指定对 Cookie 加解密的密钥。如果不指定或指定为 autogenerate，密钥将采用 Crypto API 产生的系统密钥。当指定为 autogenerate 时，产生的密钥和机器相关，因而不能跨机器和平台，除非显示给出密钥。
- loginurl：验证页面。当用户验证失败后被定向到的页面，它可以是在本地，也可以是在其他机器上。
- Cookie：指定用于验证任务的 Cookie 的名字。由于在同一台机器上可能存在多个应用，而同一应用使用一个 Cookie 名，所以同一台机器上的 Cookie 名字不能相同。

例如，在 config. Web 中有如下定义。

```
<!--config.Web-->
  <security>
<authentication mode="Cookie">
    <cookie decryptionkey="autogenerate"   loginurl="login.aspx" cookie=".ASPXCOOKIE-
DEMO"
/>
</ authentication>
</security>
…
```

从这段配置可以知道，Cookie 的密钥由系统产生，当验证失败后，页面将跳转至 login. aspx，Cookie 名为. ASPXCOOKIEDEMO。

在基于 Form 的验证中，一个常用到的对象是 CookAuthentication。

CookAuthentication 有 4 个比较重要的方法。

- RedirectFormLoginPage 方法：通常在验证成功以后，从用户的验证页面返回开始请求的页面。它带有两个参数，第一个为 Cookie 要记录的用户名，第二个表明是否记录到永久性 Cookie 中。

例如，CookAuthentication. RedirectFormLoginPage("Dan", True)表示把用户 Dan 记录到硬盘上的 Cookie 记录中，然后跳转到用户原先请求的页面上。

- GetAuthCookie：从 Cookie 中取得指定用户名的值。它的参数和 RedirectFormLoginPage 方法是一样的。

例如，CookAuthentication. GetAuthCookie("Dan", False)表示从当前连接的 Cookie 中取出用户名为 Dan 的值。

- SetAuthCookie：把指定用户名存入 Cookie 中。参数和 GetAuthCookie 方法一致。

例如，CookAuthentication. SetAuthCookie("Dan", False)表示把用户 Dan 记录到连接过程中存在的 Cookie 中。

- SignOut：删除当前用户的验证 Cookie，它不会理会到底是在内存还是在硬盘上。

2. 授权用户和角色

URL 授权控制对资源的访问权限。它可以使一些用户和角色对资源有存取权限，也可以拒绝某些用户和角色对资源的存取，甚至还可以决定能够存取资源的 HTTP 方法(例如，不允许 get，允许 POST 等)。

对于授权用户和角色的控制，ASP. NET 通过配置文件 config. Web 中的<authorization>标识段来实现。<allow>标识表示允许对资源的访问，<deny>标识表示拒绝对资源的访问。

它们都有两个属性，users 和 roles 分别表示用户和角色。

下面来看一个实例。

```
<!--config.Web-->
<configuration>
    <security>
      <authorization>
        <allow users="nobody@163.net"/>
          <allow roles="Admins"/>
          <deny users="*"/>
        </authorization >
      </ security>
    </ configuration>
```

它表明了这样一个事实，用户 nobody@163. net 和角色 Admins 有访问本站点的权限，其他用户对本站点的访问将被拒绝。也就是用户 nobody@163. net 和角色 Admins 分别是授权用户和授权角色。

同样，可以定义多个用户或者角色被授权或禁止，它们之间以“，”分隔。

例如：

```
<allow  users="Dan, Li, Wang"/>
  <deny  roles="Admins, Everyone"/>
```

表示用户 Dan，Li，Wang 是授权用户，但是角色为 Admins 或者是 Everyone 的被排除在外。它的效果和分开写是一样的，如上例也可以写为：

```
<allow  users="Dan"/>
  <allow  users="Li"/>
  <allow  users="Wang"/>
  <deny  roles="Admins"/>
  <deny  roles="Everyone"/>
```

此外，还可以决定用户的某种 HTTP 方法是否可以被允许，方法是使用 verb 属性来表明对那种 HTTP 方法操作。

例如：

```
<allow  verb=post  users="Dan,Li"/>
  <deny  verb=get  roles="everyone"/>
```

表示允许用户 Dan，Li 采用 post 方法访问资源，而拒绝角色 everyone 的 get 方式对资源的访问。

符号“*”和“?”在<allow>和< deny >标识中有特殊的含义，“*”表示任何用户，“?”表示匿名用户。

例如：

```
    <authorization>
        <allow users="*">
        <deny users="?">
      </authorization >
```

表示除了匿名用户以外的所有用户都被允许访问本站点。

配置文件也呈层次结构,这也就导致了用户和角色的授权不是单一的结果,它取决于沿树形结构上所有配置文件指定的结果的合集,而且越接近叶节点的配置越有效。

例如,访问 http:www.my.com/MyApp/a.aspx。在 http:www.my.com 根目录下的配置文件 config.Web 有如下内容。

```
<security>
<authorization>
  <allow users="*">
</authorization>
</security>
```

而在 MyApp 目录下有配置文件 config.Web,内容如下。

```
<security>
<authorization>
  <allow users="Dan">
<deny users="*"/>
</authorization>
</security>
```

那么,授权用户的集合到底是怎样的呢? ASP.NET 首先取得站点根目录下的配置,即所有用户都被允许访问;然后 ASP.NET 进入 MyApp 的目录并取得其下的配置,除用户 Dan 以外所有的用户被拒绝;最后合并两个授权集合,并且如果两者之间有冲突,以后者为准。所以最终的授权集合为用户 Dan,其他用户被拒绝。

7.2　网站的维护

网站建好后并不是一劳永逸的,还需要精心地运作才会显现成效。网站是企业虚拟的办公地点,所以理所应当有人看管,网站管理员就是监守网站这个虚拟办公室的人员。他是网站的用户和合作伙伴等一切网站来客的联络人,同时又肩负着这个虚拟办公室的日常运营及维护的重任。网站管理员,顾名思义就是管理网站的人。就网站的特殊性而言,它好比是企业运营的电子版,只有不断地进行建设、维护和更新,才能与企业的发展同步。网站建成以后一定要设置网站管理人员,它不仅包括 Webmaster,而且包括所有的技术人员和网页日常维护人员。

7.2.1 设置网站管理员

1. 设置网站管理员的必要性

1）能及时与网站来客保持联系

一些企业网站设立后，网站所设置的电子信箱一直没有正常使用或多人使用，没有发挥电子邮箱应有的作用。要知道，用户访问了网站后，如对某些方面感兴趣，往往会写个留言、发封邮件。这些网站来客就很可能是潜在的客户群，应该给予必要的重视，对他们提出的问题及时解决并回复。这不仅有助于企业建立良好的公众形象，而且能进一步增加网站回头客的数量。反之，如果没有专人负责处理网站来客邮件信息的话，这些网站来客的留言或电子邮件不是发不出去就是发信后石沉大海，迟迟得不到回复。而且根据经验，尽管网站上有地址、电话和传真号码等，但访问者还是喜欢用电子邮件联系，因为网站本身就是突出这种通信优势。所以要保持网站与访问者能及时沟通，就必须设置网站管理员。

2）能对网站进行日常的技术管理

一些网站在建站初期，未能将信息全部组织到位，这种情况是很常见的。关键是要在建站之后，不断对网站的内容进行更新和补充，并且维护好历史资料。如果过了很长一段时期后依然内容干巴巴，从不更新，甚至链接断层、界面零乱、文件丢缺，一派衰落景象的话，只会使辛辛苦苦建起的网站变成垃圾，也就谈不上对客户提供必要的服务了。因此网络不能被孤立地对待，必须将其作为企业的一个部分。对于一个信息化的企业来说，网络就是业务，人们必须从业务的角度来看待网络管理，高效率的网络管理手段是业务成长的保障。

3）能对网站安全进行监控

网站是对外开放的，每个来客都能进行访问，因此要注意网站的安全问题。要防止某些心怀不轨的来客对网站进行攻击，修改网页搞恶作剧，破坏系统程序或施放病毒使系统陷入瘫痪，盗用服务器磁盘空间建立自己的个人主页或兴趣站点，传播黄色、反动信息，窃取政治、军事、商业秘密，进行电子邮件骚扰及转移资金账户等。这就需要网站的管理员对网站进行严密的防护，设置防火墙，采用加密算法进行密钥传输和用户身份认证，并能及时发现网站的漏洞予以处理。

寻找网络中的薄弱环节和安全漏洞是每个系统管理员和每个黑客都在做的一件事。系统管理员只有能够检测并发现它的所在，才有可能控制它，才能领先黑客一步。另一方面，网络是动态的，黑客也是多谋善变的。买安全产品或服务，仅配置一次是不够的，防火墙如此，其他安全产品也是如此。

2. 网站管理员的设置原则

网站管理员的设置原则是多种多样的，有的是按照板块设置管理员，如财经板块、体育板块等，这种设置适用于内容丰富全面的大型站点。有的是根据技术差异角度设置管理员，如系统管理员、数据库管理员和普通管理员等，这种设置适用于对安全性要求较高的站点。现行的做法是多样的，最根本的一个原则就是要结合网站管理的需要设置管

理员。

3. 确立制度

确立和完善管理制度是网站管理的客观要求。网站管理员只有在明确其工作职责、工作内容及需要掌握的相关技能后，才能有的放矢地开展工作，以达到良好的效果。

4. 明确工作职责

综上所述，网站管理员的工作是非常重要的，所以管理员要认真履行各项规章制度，对所管理的工作抓好抓实。同时管理员有义务为网站的发展和规划提供良好的建议。网站管理人员的主要责任是：进行网站的更新维护，与企业各有关部门紧密联系，及时编辑和组织网站相关信息内容，保持网站处于一种活力状态下而成为信息反馈的中枢，每天检查网站的运作情况和通信情况并及时处理。

7.2.2　网站维护的主要内容

网站的管理维护主要包括检测网站的错误、保证网站正常运转、处理用户信息、定期更新网页内容和修正网页错误、确认网页的程序和链接的正确性、跟踪访问者的信息，清楚网站的点击状况、备份动态数据库等。对于公司、企业等单位，尤其是拥有自己服务器的单位，则需要配置专门的网站管理员。

企业网站中最常见的更新包括公司新闻的发布、新产品的推出、公司举办的各种名目的活动及网上答疑等，这些都需要在公司站点上实时体现。当然，网站的维护工作还有许多，不过最主要的目的还在于评估所建网站的运行情况，及时知道访者的反馈信息，分析情况，以便为公司的发展及网站的运营作为参考依据。

在 Web 站点上履行维护工作到底是谁的责任，这个问题很难回答。大公司经常把工作转包给顾问或其他公司，小公司有时把任务加到现有雇员的职责中。对一般用户而言，可以雇用一个公司集中完成 Web 站点上特定的维护工作。

1. 网站内容的维护和更新

网站的信息内容应该适时地更新。在网站栏目的设置上，最好将一些可以定期更新的栏目如企业新闻等放在首页上，使首页的更新频率更高些。此外，当网站变得十分庞大时将会有不计其数的图片和网页文件等内容，如果它们有一个丢失或链接失败都会引起网页错误，所以一定要保证整个网站的“健康”和完整。这就是狭义的维护概念。网页更新要做到以下几点。

1）专人维护新闻栏目

这是非常重要的。一方面把企业、业界动态都反映在里面，让访问者觉得这是一个发展中的企业；另一方面，要在网上及时收集相关材料放置到网站上，以引起同类用户的兴趣。

2）时常检查相关链接

通过测试软件对网站所有的网页链接进行测试，看是否能连通，尤其是网站导航栏目可能经常出问题。最好是自己亲自浏览，这样才能及时发现问题。也可以在网页上显示“如有链接错误，请指出”等字样。

2. 定期检测网站、保证服务器的正常工作

这是管理网站最基本的工作之一。如果是自己公司的服务器，就需要保证服务器的正常工作，保证网络的正常运转，保证用户可以正常地访问网站并使用网站上提供的各种服务等，这需要懂得一定技术的人员来担任网站管理工作。如果是租用的虚拟主机或个人申请的免费网页空间，则只需要定期检测网站，保证正常运转，检查并修正网页错误即可。

按照网站的规模大小，这些工作都需要借助一些工具软件来完成，读者可参考一些这方面的其他书籍。

这是所有管理维护工作的基础，具有良好的稳定性是一个网站生存的必要条件。试想一下，一个总是包含错误的链接，导致网页无法打开，或者总是跳出烦人的页面错误窗口的网站，会有持久的生命力吗？如果这是公司的网站，则会严重影响到公司的形象，用户不但会不满意公司的服务质量，甚至会怀疑公司的实力，公司会因此失去不少的用户群。

因此，在保证网站畅通无阻、正常运作的前提下，才可进一步地运行网站的维护和改进。

3. 处理网站数据和用户信息

网站建设好以后，下一步的工作就是研究如何发展网站，在这里充分利用一些网站数据和用户的反馈信息，可以使工作达到事半功倍的效果。

1）处理、分析网站数据

在构建网站时，可以在网页上加载计数器或者使用访问日志，这些都有利于帮助分析网站的情况以便于进一步的发展。

计数器是用来统计访问网页人数的工具，在首页上放置计数器即可统计出网站的访问量。许多提供虚拟主机、免费空间的网站都提供计数器，也可以到一些提供统计服务的网站获得这项服务。下面是几个提供计数统计服务的网站。

- 天堂免费计数器(http://www.xcinfo.ha.cn/jsq/login.asp)
- FoxWeb(http://www.foxweb.com)

许多提供计数统计的网站除了能够提供访问量外，还能够提供开始统计日期、总统计天数、总浏览人数、日访问量、预计本日浏览人数、各栏目访问量、站点性质和总排名等服务，可以根据这些数据及时调整网站的各个栏目，制定下一步的网站发展计划等。

访问日志是用来记录用户在网站进行各种操作的，最初的访问日志是用来维护简单的用户请求信息，后来经过逐步扩展，包含了诸如浏览者来自哪里、浏览者使用的浏览器类型、用户最先浏览的网页、用户最后浏览的网页、访问最多的网页、访问最少的网页和下载次数最多的文件等。这些数据会对管理者要做的分析带来很大的帮助，这样可以有计划地调整网站，哪些地方应该加强，哪些地方应该删除等。对访问日志的数据分析一般使用专门的软件，如 WebTrends Log Analyzer 等。

2）处理用户信息

通常一个网站建好后，网站管理者除了要日常维护站点之外，还必须与访问者多沟

通。仅仅有精美的网站设计、先进的技术应用及丰富的内容，访问量不一定会上升。在产品销售生产中，一般应先做市场调查，看看市场上人们需要什么样的产品，然后才开始生产。产品推出后，还要收集反馈信息，对产品进行改造，进一步迎合客户的需要。站点建设也同产品销售一样，必须同站点访问者多交流，这样才能了解网民需要看什么。如果不了解访问者的想法，只是按照自己的思路去发展，那么站点只能是空中楼阁。

企业应设专人或专门的岗位从事网站的服务和反馈处理工作。客户向企业网站提交的各种反馈表单、购买的商品、发到企业邮箱中的电子邮件，以及在企业留言板上的留言等，企业如果没有及时处理和跟进，则不但会丧失机会，还会造成很坏的影响。

因特网之所以能够迅猛地发展，除了它能为用户带来海量的信息外，它与传统媒体的最大区别之一就是它具有和用户的交互性，使得网站能够为用户制作许多个性化的服务。一个具有精美的网站设计、先进的技术应用和丰富的内容更新的网站，不见得访问量就一定能够上升。如果能够在此基础上，重视用户信息的反馈，及时与用户交流、沟通，这样给用户的感觉会更具有亲和力，也更有利于提高网站的“回头率”。如果用户发现他的意见或建议被网站采纳，或者感觉到网站对他的关注，就不但会经常光顾网站，甚至可能会成为网站的义务宣传员。网站维护与反馈工作主要体现在以下这些方面。

(1) 用户留言。

建立了留言板就要经常查看，在留言板中留言的用户一般都是对网站感兴趣的用户，大多客户会在这里留下对网站的意见、建议，尤其是一些无法显示的网页错误等，提醒管理者及时纠正、解决，有的客户会对网页外观，他所需求的信息内容及更新时间等提出自己的要求，这对网站的日后更新、升级都是值得借鉴的资料。

制作好留言簿后要经常维护，收集意见。因为一般访问者对站点有什么意见，通常都会在第一时间看看站点哪里有留言簿，然后就在那里记录，期望网站管理员能提供其想要的内容，或提供相关的服务。所以必须对别人提出的问题进行分析总结。一方面要以尽可能快的速度进行答复，另一方面也要记录下来进行切实的改进。这样可以从中收集到很多信息，获得很多商机。

(2) 用户电子邮件。

一般情况下，网站在首页上都会留下与网站主人联系的方式，这样会方便客户与网站联系。对于用户的来信应该及时回复，有些用户会在信件中提出他们自己的要求和问题，管理者应该做出认真的解答。重视每一封来信，争取每一个回头客，这就像做生意一样，回头客对访问量的增加很重要。如果长时间对用户的信件置之不理，就会失去一批难得的回头客。网站逐渐做大后，可以设专人管理，对于固定格式信件可设置自动回复。

所有的企业网站都有自己的联系页面，通常是管理者的电子邮件地址。经常会有一些信息发到邮箱中，对访问者的邮件要及时答复。最好是在邮件服务器上设置一个自动回复的功能，这样能够使访问者对站点的服务有一种安全感和责任感，然后再对用户的问题进行细致的解答。

(3) 电子邮件列表。

电子邮件列表是很多企业网站上都有的，对电子邮件进行维护也很重要。一方面要保证发送的频率，另一方面要保证邮件的内容，要有新意，而且最好能与收集的意见相结

合。总之了解和关心网站的访问者越多，就越有利于站点的发展。

(4) BBS。

如果网站有 BBS 论坛，同样应该重视在这里发布的文章。因为 BBS 空间的自由度较大，除了要注意有用的信息，更应该注意要及时删除包含不健康、反动、有恶劣影响、漫骂他人及发布垃圾广告等类型的文章，以保证网站健康、清洁的形象。

BBS 是一个自由的天地，作为企业网站可以在其上自由地讨论技术问题，而对于 BBS 的实时监控尤为重要。例如对一些色情和反动的言论要马上删除，否则一来会影响网站的形象，二来可能会引起大麻烦。从另一个角度讲，BBS 中也可能会出现一些乱七八糟的广告，对此要及时删除，否则影响 BBS 的性质，不会再吸引浏览者。有时甚至会出现一些竞争对手的广告或诋毁企业形象的言论，更需要及时删除。同时要多收集一些相关资料在 BBS 中发表，以保证 BBS 园地的学术性质。

(5) 投票、调查问卷。

每隔一定的时间，还可以在网站上发布一些投票、调查问卷之类的活动，以吸引用户的参与。不管是针对哪一方面的调查，都应该精心设计，例如，抓住一些社会热点话题，从独特的视角反向提出问题，使问题具有较强的针对性和个性，引起用户的兴趣。

企业站点上经常会有一些投票调查的程序，用来了解访问者的喜好或意见。一方面要对已调查的数据进行分析，另一方面也可以经常变换调查的内容。但对于调查内容的设置要有针对性，不要搞一些空泛的问题。也可以针对某个热点投票，以吸引别人关注结果。

4. 定期更新、动态维护

管理维护网站容易产生的误区就是以为保证网站的正常运转就万事大吉了。其实不然，应该不断地宣传以吸引新用户的光临，更重要的工作是要不断地维护、经常地更新，以保证老顾客的需求，这才是一个网站的立身之本。一个总给人以新意、具有不断充实的新内容和信息的网站，才是真正具有生命力的网站。

网站的更新应该在处理客户信息的基础上，根据统计分析的结果，及时摸清大多数用户的兴趣所在，有针对性地进行更新。

要有规律地更新维护网站，定时维护，这样既节省时间，而且会减轻劳动强度，因为有规律，更方便网友们查阅和使用。

当我们有了创意要及时维护，珍惜难得的灵感和创造力。对用户、网友提出的较集中的问题，及时做出反映。做到动态维护、定期更新。

更新的内容要充实、及时，不要让访问者感觉到是在应付。这就需要平时丰富的积累和广泛地搜集材料。

进行更新维护的具体工作就是进行网页文件的上传、下载、更改和删除等。

在这里还需要注意的一个问题就是对历史内容的存放处理。网络服务器的空间毕竟有限，尤其是提供申请租用网页空间的网站对网页空间大小都有限制，对于网站上有保存价值的信息数据应该及时处理。

5. 修正网页错误

这也是网站管理维护的基本工作，有了前面的这些基础，就应该能找到网站中有问

题、有错误的网页了。

修正网页的错误，首先包括网页上的技术性错误。这些错误包括网页的链接错误、网页无法正常显示、无法看到图片、无法听到背景音乐、过大的网页、过大过小的字体，前景色、背景色色差过小以至于看不清等。这些错误需要回到设计和制作网页时使用的工具中进行再次的调整、编辑，然后再上传到服务器。

还有就是网页中的内容错误。除了用户指出的错误外，管理者也应该经常关注自己的网站，从中查找错误，尤其是一些介绍技术或技术性产品的网站，以免给用户不好的印象。

6. 网上推广与营销不可缺少

要想让更多的人知道自己的网站，就要在网上进行推广。网上推广的手段很多，大多数是免费的。主要的推广手段包括搜索引擎注册、注册加入行业网站、邮件宣传、论坛留言、新闻组、友情连接、互换广告条及 B2B 站点发布信息等。除了网上推广外，还有很多网上与网下结合的渠道，例如可将网址和企业的商标一起使用，通过产品、信笺、名片和公司资料等途径可以很快地将网站告知用户，以方便用户从网上了解企业的最新动态。

7. 不断完善网站系统，提供更好的服务

初建网站一般投入较少，功能也不是很强。随着业务的发展，网站的功能也应该不断完善以满足用户的需要。此时使用集成度高的电子商务应用系统可以更好地实现网上业务的管理和发展，从而将企业的电子商务带向更高的阶段，取得更大的成绩。

8. 网站备份

网站维护最后要注意的就是对网站文件的备份了。如果服务器发生了灾难，网站就很有可能被彻底毁坏，因此时常备份网站文件是非常重要的。

7.2.3　选择站点维护工具

如果决定自己完成站点维护，可以利用许多有用的工具。要改变 HTML，当然需要某种 HTML 编辑器，许多 HTML 编辑器提供站点维护的专用工具，如链接和 HTML 验证器，因此也要确保选择这些工具。另外，不要忘记与主机公司取得联系，因为许多公司提供主机软件包的维护工具。下面将简单介绍两种主要的用户定制维护工具。

1. 改进工具

在改进的维护工具中，服务包括从链接验证器到信息量分析器的全部范围，例如，Netscape 公司的 Web SiteGarage's Tune Up 服务有以下特点。

(1) 浏览器兼容性检查。检查页面是否能在不同的浏览器版本中显示。

(2) 搜索引擎可读性。确认页面包括正确的 META 标记。

(3) 下载时间检查。报告页面下载速度的快慢程度。

(4) 链接检查。找出过时的链接。

(5) 链接普及性检查。确定有多少页面用流行的搜索引擎链接到站点。

(6) 拼写检查。指出拼错的词。

(7) HTML 检查。检查页面是否符合 W3C HTML 的规范。

许多其他公司现在也提供类似的服务,下面列出了一些受欢迎的公司。

- Doctor HTML(www2.imagiware.com/RxHTML/)
- LinkAlarm(www.linkalarm.com)
- Net Mechanic(www.netmechanic.com)

迄今为止,我们列出的这些工具都是在线运行的,用户可以购买几个质量好的产品安装到自己的系统中。一般来说,Web 管理员更愿意使用下面的这些产品来管理和维护他们自己的站点。

- CyberSpyder 的 Aman Software(www.cyberspyder.com)
- Linkbot 的 Watchfire(www.watchfire.com/products/linkbot.htm)
- WebMaster 的 Coast Software Inc.(www.coast.com)
- WebTrends 的 WebTrends 公司(www.webtrends.com)

2. 服务器监视

一种较新的联机维护工具是第三方(third-party)的服务器监视。有些公司(例如@Watch 或 Server Rat 等)对 Web 服务器进行监视,定期检查用户服务器,以确保所有的工作都在正常地进行。如果他们检查出故障,会在 5 分钟之内通知用户。

这种服务最大的优点是监控器本身是独立的,因为监控公司不是主机公司,用户更希望能把服务中的失误及时地通知他们。

参 考 文 献

1. 姚怡，余海萍. 网站建设与管理. 北京：中国铁道出版社，2005
2. 陈明. 数据库系统及应用——SQL Server 2000. 北京：清华大学出版社，2007
3. 邵丽萍，王馨迪，陆军. ASP 动态网页设计. 北京：中国铁道出版社，2006
4. 蔡立军，池鹏. 网站建设原理与实践. 北京：中国水利水电出版社，2005
5. 邵丽萍. 网站开发实践实验教程. 北京：电子工业出版社，2007

读者意见反馈

亲爱的读者：

感谢您一直以来对清华版计算机教材的支持和爱护。为了今后为您提供更优秀的教材，请您抽出宝贵的时间来填写下面的意见反馈表，以便我们更好地对本教材做进一步改进。同时如果您在使用本教材的过程中遇到了什么问题，或者有什么好的建议，也请您来信告诉我们。

地址：北京市海淀区双清路学研大厦A座602室　计算机与信息分社营销室 收
邮编：100084　　电子邮件：jsjjc@tup.tsinghua.edu.cn
电话：010-62770175-4608/4409　　邮购电话：010-62786544

教材名称：网站建设实用教程
ISBN：978-7-302-18055-5

个人资料

姓名：________ 年龄：______ 所在院校/专业：____________

文化程度：________ 通信地址：____________________

联系电话：________ 电子信箱：____________________

您使用本书是作为： □指定教材 □选用教材 □辅导教材 □自学教材

您对本书封面设计的满意度：

□很满意 □满意 □一般 □不满意　改进建议____________________

您对本书印刷质量的满意度：

□很满意 □满意 □一般 □不满意　改进建议____________________

您对本书的总体满意度：

从语言质量角度看 □很满意 □满意 □一般 □不满意

从科技含量角度看 □很满意 □满意 □一般 □不满意

本书最令您满意的是：

□指导明确 □内容充实 □讲解详尽 □实例丰富

您认为本书在哪些地方应进行修改？（可附页）

____

____

您希望本书在哪些方面进行改进？（可附页）

____

____

电子教案支持

敬爱的教师：

为了配合本课程的教学需要，本教材配有配套的电子教案（素材），有需求的教师可以与我们联系，我们将向使用本教材进行教学的教师免费赠送电子教案（素材），希望有助于教学活动的开展。相关信息请拨打电话 010-62776969 或发送电子邮件至 jsjjc@tup.tsinghua.edu.cn 咨询，也可以到清华大学出版社主页（http://www.tup.com.cn 或 http://www.tup.tsinghua.edu.cn）上查询。